Purdue Model **Kubeflow** **Monitoring**

ADMIN
Network & Security

ISSUE 85

Monitoring
Networks, applications, and long-term storage

Bloonix
Web-based network monitoring

Kubeflow
Kubernetes setup for AI development

Azure Dev Box Cloud Service

Prometheus Long-Term Storage

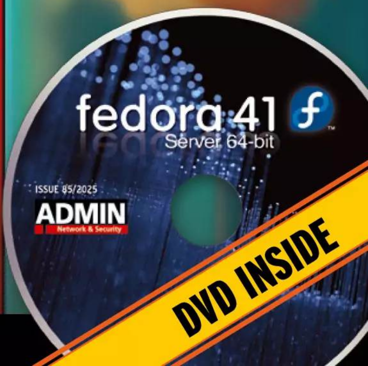
Attack Surface Management
Tools and strategies for threat management

Podman Quadlets
Run rootless containers as systemd services

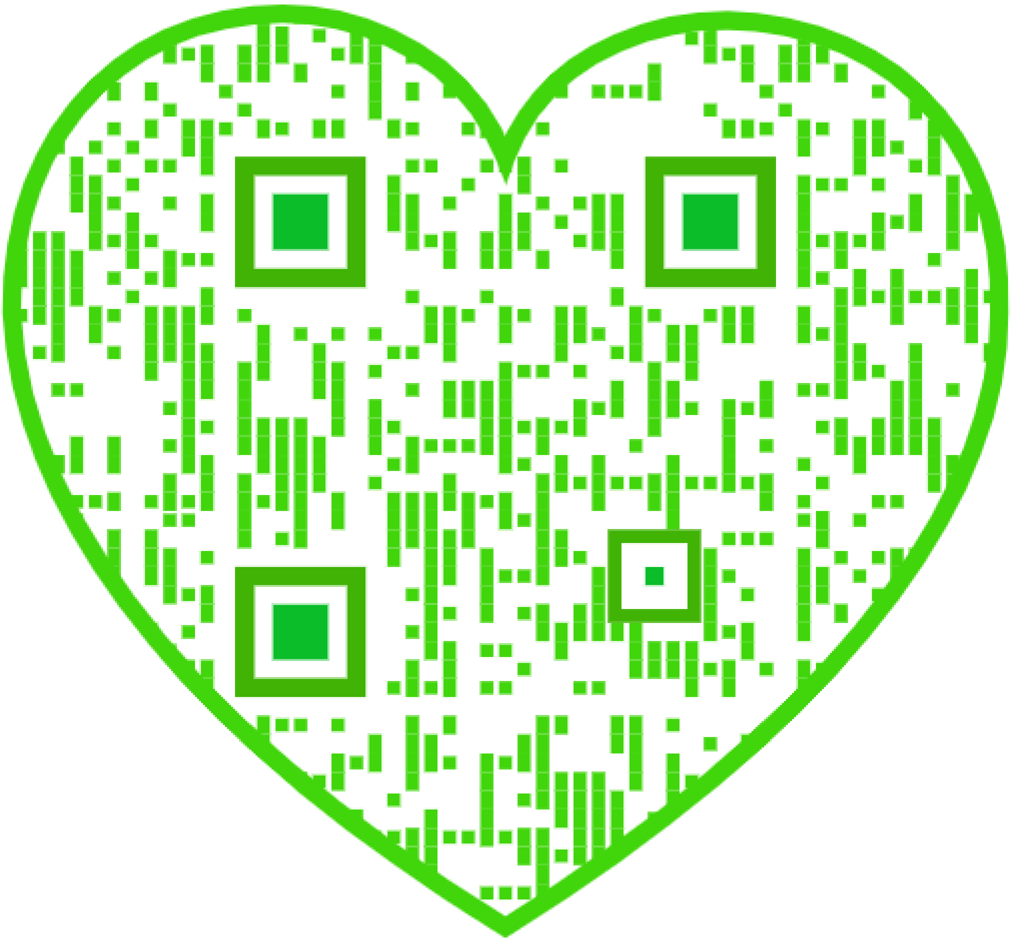
Purdue Model
Map the challenges of industrial systems

Julia
A backbone for interactive websites

LINUX NEW MEDIA
The Pulse of Open Source


fedora 41
Server 64-bit
ISSUE 85/2025
ADMIN
Network & Security
DVD INSIDE

WWW.ADMIN-MAGAZINE.COM



BLACK MYTH
WUKONG

Linux compatible

Up to 5 Years Guarantee

Immediately ready for use

TUXEDO

tuxe.do/lxadmin85

Made in Germany

German Data Privacy

German Tech Support

Looking Backward, Looking Forward

Whatever IT challenges you face in 2025, will you meet them with anger, fear, and indifference – or will you go forward with interest, curiosity, and humility?

A new year always brings the hope of new possibilities, new technologies, and new challenges. I'm not sure that new challenges are what system administrators want to look forward to, but you know they're there and they're coming whether you like it or not. Challenges, in one way or another, are what you do. You face challenges and you resolve them. I remember quite well when Windows XP (XP) hit my clients' desktops and I was thinking, "Well, I guess I can get out of the computer business because XP won't have all the issues of its predecessors." It seemed like the perfect operating system – the best since OS/2.

I was wrong, of course, because I wasn't a regular user. Regular users work with many different applications and with the inherent bugs, user error, and usual problems associated with any desktop conglomeration. I was still in business facing the challenges that XP presented to my clients. XP was good, but it wasn't perfect, and once my client base latched onto XP, they didn't want to let it go. Thank you, Microsoft.

I no longer support desktop operating systems. My focus has long since shifted to the data center and then to the cloud. It's been more than 15 years since I labored in the trenches of desktop support, which is the front line of the IT world. I still interface with clients daily, but the stress of having someone watch me troubleshoot while telling me they "haven't changed anything" is something I don't miss. I'll leave that realm to the younger, more patient souls among us.

These days I'd rather deal with the cold blinking cursor that has nothing to say. Working remotely, I can express my frustrations out loud without the fear of someone reacting badly to my bad reactions. Yes, I still get upset when I realize that spelling counts and that computers don't listen well to verbal commands (yet). Support, in any location or setting, is frustrating – even when dealing only with that blinking cursor. Alas, I digress.

The challenges you will face in 2025 are different from those you've faced previously. Now, you must face not only the challenge of management calling for more automation, but also the move toward artificial intelligence (AI) guiding that automation. Admittedly, AI is cool for some things (creating funny pictures, making short but off-the-wall film clips, and writing stories), but after a few minutes, the non-artificial reality sets in.

I can foresee systems rebooting randomly; patching events going wrong, with multiple systems never recovering; and user accounts (e.g., administrator and root) being locked out so that no human intervention or mitigation is possible. Maybe I'm paranoid. OK, yes, I'm definitely paranoid, but my paranoia is justifiable because no matter how good one believes a system or process is, something will always go awry. When it does, your users better hope they have some experienced system administrators around to fix the issues and return everything to normal.

Automation and AI aren't the only challenges moving forward. You still have users, operating systems, hardware, Internet outages, updates, upgrades, maintenance, vendor support, and human shortcomings to endure. You will always have challenges to face. They might change over time, but what remains the same is your approach to the challenges you meet.

From experience, I can tell you that difficulties are far less daunting if you have a good attitude. It's true. A calm, systematic process is what you need to achieve success. It is the non-emotional approach to problem-solving that makes automation and AI so attractive to those who hear nothing but complaints from users and IT staff. Who wouldn't want to use a robot or virtual system administrator? Very few users ever come away from a conversation with an IT person feeling better than before. However, submitting a question to a chatbot that will solve your problem without a negative response, sigh, or eyeroll is a breath of fresh air. Keep these things in mind as you move forward.

Ken Hess • ADMIN Senior Editor

Image © thvideo, 123RF.com

ADMIN
Network & Security

Features

12 Bloonix
Integrate all the tools you need to monitor large IT infrastructures 24x7 to prevent failures.

18 End-to-End Monitoring
Your web server is running perfectly according to Nagios, but is it delivering the intended user experience? End-to-end monitoring lets you know whether your application is performing as expected.

24 Prometheus
If you use Prometheus as a time series database, you will know that the more data it stores, the slower it becomes. Thanos, Cortex, Mimir, and M3DB set out to solve this problem in totally different ways. We reveal the candidates' strengths and weaknesses.

Tools

30 Automating Microsoft Dev Box
Explore automation techniques for the Azure Dev Box cloud service.

38 Talos
In the world of container virtualization, the operating systems of compute nodes are largely degraded to non-player characters that can do little more than start and stop containers. This lean Linux distribution for Kubernetes takes the game to the extreme and offers a system that weighs in at less than 90MB.

44 Kubeflow
We walk you through a Kubernetes setup for AI development on an AWS account.

50 mitmproxy
This HTTPS proxy puts interactive traffic analysis in your hands.

Containers and Virtualization

56 Podman Quadlets
Running rootless containers is easy with Podman. With quadlet files, these containers are seamlessly integrated into systemd services.

62 Julia
This technical programming language is fast becoming a favorite with scientists and engineers of all stripes, but it is also well suited to form the backbone of interactive websites that explain these science and engineering concepts.

Security

72 Attack Surface Management
The tools used in ASM help identify attack surfaces more precisely and respond to changes in risk situations.

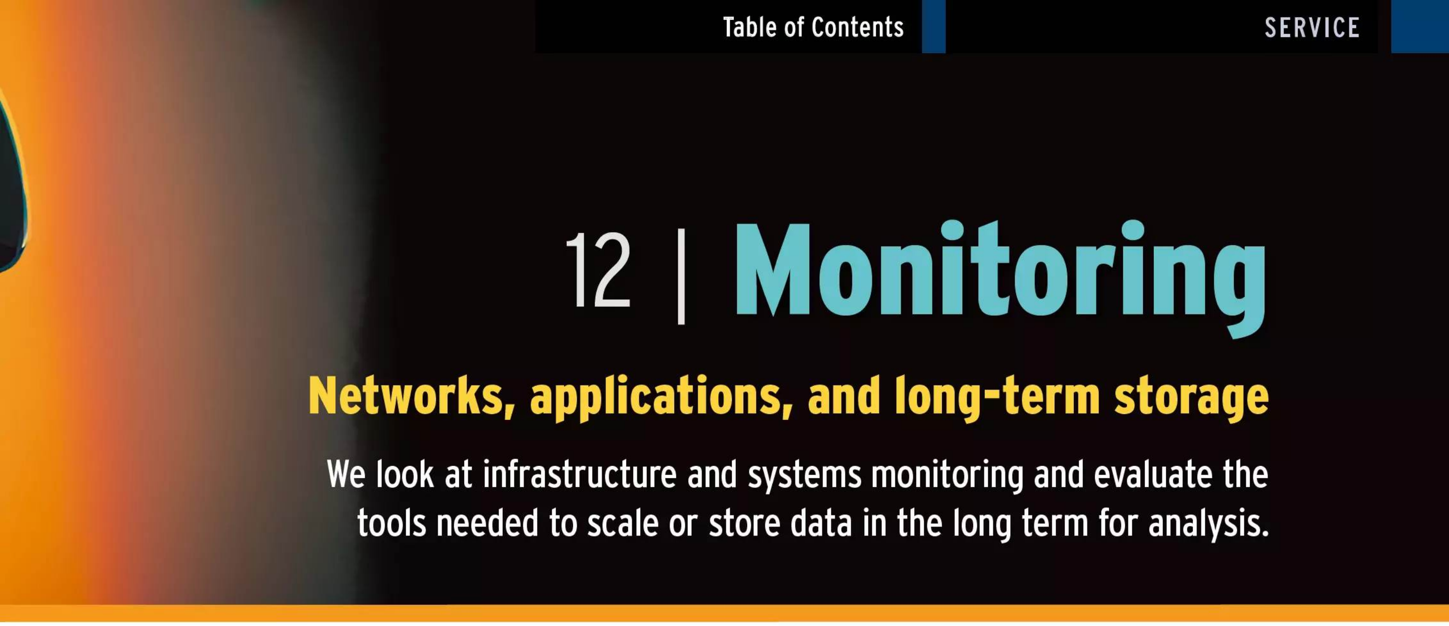
News

6 News

9 SC24
We'll recap some noticeable and not so noticeable points of interest that came up during the largest, most attended Supercomputing Conference yet.

Service

3 Welcome
96 Back Issues
97 Call for Papers
98 Coming Next Month



12 | Monitoring

Networks, applications, and long-term storage

We look at infrastructure and systems monitoring and evaluate the tools needed to scale or store data in the long term for analysis.

Highlights

30 Microsoft Dev Box
Instead of a classical VDI solution, consider a Microsoft Dev Box with project-specific templates and shared configurations that eliminate the traditional challenges of provisioning and maintaining environments.

44 Kubeflow
Training language models and AI algorithms requires a powerful infrastructure that is difficult to create manually. Kubeflow promises a remedy, and we show you an approach that lets you get it up and running fairly quickly.

78 Purdue Model
This framework maps the challenges of networking industrial systems to five levels, so you can target and mitigate risk and address vulnerabilities. We investigate an implementation tool and explain the role of zero trust.

Management

78 Purdue Model
Abstract a complex corporate network into a tangible image of individual components that simplifies the planning and implementation of protection campaigns, promotes communication, and assists in the distribution of responsibilities.

Nuts and Bolts

82 Windows Hardware Diagnostics
Seven tools for hardware diagnostics on Windows: one Windows on-board tool, five free tools, and a commercial tool.

86 Keras
A great way to start writing code with AI is to use this open source, easy-to-learn library that can use multiple frameworks.

92 Performance Dojo
Unveil the little-known superpowers of microSD cards.

On the DVD

Fedora Server 41
Installed on bare metal or virtual machines, this stable, flexible, and adaptable community server OS provisions digital services and information for organizations and individuals.

The server edition always offers the latest software with its biannual release cycle. The Fedora community counts on its reliability, stability, and security. In this new version, you can look forward to:

- Self-encrypting drives supported in the installer
- GNU toolchain update
- fedora-repoquery, a new, small command-line tool for querying repositories
- dm-vdo (virtual data optimizer) device mapper target for inline deduplication, compression, and thin provisioning
- Concurrent availability of all supported Kubernetes RPMs

fedora 41
Server 64-bit
ISSUE 85/2025
ADMIN
Network & Security
DVD ROM

@admin-magazine.bsky.social
@adminmagazine
@adminmag
ADMIN magazine
@adminmagazine

News for Admins

Tech News

Hetzner Announces S3-Compatible Object Storage

Hosting and cloud provider Hetzner has announced Object Storage — a high-performance solution for storing unstructured data — which is suitable for backing up, restoring, and archiving data, as well as for other data-intensive workloads.

The GDPR-compliant solution offers:

- S3 compatibility
- Scalability
- Security
- Cost-effectiveness

Object Storage also allows seamless integration with backup software. According to the announcement (https://www.hetzner.com/press-release/object-storage/), “data cannot be deleted and is not susceptible to ransomware attacks, and backups cannot be read or restored by unauthorized users thanks to server-side encryption.”

Hetzner’s Object Storage has a monthly base price of $5.99, which includes 1TB of storage and 1TB of egress traffic.

Read more at Hetzner: https://www.hetzner.com/storage/object-storage/.

Ongoing Cyberattack Prompts New CISA Guidance for Communications Infrastructure

US and international security agencies have warned that Chinese state hackers (https://www.bleepingcomputer.com/news/security/white-house-salt-typhoon-hacked-telcos-in-dozens-of-countries/) “have compromised networks of major global telecommunications providers to conduct a broad and significant cyber espionage campaign.”

“The PRC-affiliated cyber activity poses a serious threat to critical infrastructure, government agencies, and businesses,” said CISA Executive Assistant Director for Cybersecurity Jeff Greene. “Along with our US and international partners, we urge software manufacturers to incorporate Secure by Design principles into their development lifecycle to strengthen the security posture of their customers. Software manufacturers should review our Secure by Design resources (https://www.cisa.gov/securebydesign) and put their principles into practice.”

In light of this threat, the agencies — including CISA, FBI, NSA, and others — have jointly released a detailed new guidance (https://www.cisa.gov/resources-tools/resources/enhanced-visibility-and-hardening-guidance-communications-infrastructure) document to help network administrators and defenders identify anomalous behavior, improve configuration, harden their devices, and limit the attackers’ access.

For example, the device hardening section for network engineers includes actions such as:

- Use an out-of-band management network that is physically separate from the operational data flow network. Ensure that management of network infrastructure devices can only come from the out-of-band management network.
- Implement a strict, default-deny ACL strategy to control inbound and egressing traffic and ensure all denied traffic is logged.

Get the latest IT and HPC news in your inbox

Subscribe free to ADMIN Update and HPC Update

bit.ly/HPC-ADMIN-Update

Lead Image © vlastas, 123RF.com

- Employ strong network segmentation via the use of router ACLs, stateful packet inspection, firewall capabilities, and demilitarized zone (DMZ) constructs.
- Place externally facing services, such as DNS, web servers, and mail servers, in a DMZ to provide segmentation from the internal LAN and back-end resources.
- Do not manage devices from the Internet. Only allow device management from trusted devices on trusted networks.
- Control access to device Virtual Teletype (VTY) lines with an ACL to restrict inbound lateral movement connections.
- If using Simple Network Management Protocol (SNMP), ensure only SNMP v3 with encryption and authentication is used, along with ACL protections against unnecessary public exposure.
- Disable all unnecessary discovery protocols, such as Cisco Discovery Protocol (CDP) or Link Layer Discovery Protocol (LLDP).
- Disable Internet Protocol (IP) source routing.
- Disable Secure Shell (SSH) version 1. Ensure only SSH v2.0 is used with the following cryptographic considerations. For more information on acceptable algorithms, see NSA's Network Infrastructure Security Guide: https://media.defense.gov/2022/Jun/15/2003018261/-1/-1/0/CTR_NSA_NETWORK_INFRASTRUCTURE_SECURITY_GUIDE_20220615.PDF.

Organizations that believe they are a victim of these attacks should contact their local FBI Field Office or CISA. Read the complete list of guidelines at CISA: https://www.cisa.gov/resources-tools/resources/enhanced-visibility-and-hardening-guidance-communications-infrastructure.

OpenMP 6.0 Released

The OpenMP Architecture Review Board (ARB) has released version 6.0 of the OpenMP API Specification.

According to the announcement (https://www.openmp.org/home-news/openmp-arb-releases-openmp-6-0-for-easier-programming/), this major upgrade makes it easier to develop parallel programs and offers more fine-grained control to developers.

Updates include:

- Simplified task programming
- Enhanced device support
- Support for induction
- Support for parallelization of the latest C, C++, and Fortran language standards
- Greater user control of storage resources and memory spaces

"The OpenMP API is used in a range of fields including physics, automotive and aeronautical simulations, biotech (including drug design), automation, robotics and financial analysis. It covers the entire hardware spectrum from embedded and accelerator devices to multicore systems with shared-memory," the announcement states.

To learn more, check out the educational videos on the OpenMP YouTube channel (https://www.youtube.com/playlist?list=PLLX-Q6B8xqZ8n8bwjGdzBJ25X2utwnoEG) or visit the OpenMP website (https://www.openmp.org/).

Open Source Development Improves Software Security, Says LF Report

Eighty-one percent of open source developers think that open source development leads to code that has better quality and security, according to the 2024 Open Source Developer Report (https://www.linuxfoundation.org/research/open-source-developer-survey-2024) from the Linux Foundation.

This year's report focused on "the specific needs and strategies that open source developers employ to advance their careers," but it also asked developers about the overall value of open source software (OSS).

Those findings include:

- 88 percent of respondents somewhat or strongly agree that open source software (OSS) allows organizations to innovate faster.
- 69 percent agree that the overall benefits of open source outweigh the cost.
- 49 percent agree that open source software has sustainability problems.
- 82 percent agree that open source sustainability problems could be solved if organizations did a better job of "giving back."

In regard to career advancement, the report found that “59 percent of respondents perceived value in keeping up with open source ecosystems by attending professional events.”

Other strategies include:

- Learning new skills (87% of respondents found this very or extremely important)
- Improving existing skills (87%)
- Demonstrating excellence in current role (81%)
- Staying up to date with fast-paced technology changes (77%)
- Understanding how technology will impact the industry I’m employed in (77%)
- Connecting to other professionals (60%)
- Changing roles or employers (28%)

See more details in the 2024 Open Source Software Developer Report: https://www.linuxfoundation.org/research/open-source-developer-survey-2024.

Most Organizations Are Unprepared for Climate-Related Disruptions

Only 20 percent of organizations surveyed have identified climate-related risks across their operations, according to the latest report from ACCA (https://www.accaglobal.com/gb/en/professional-insights/global-profession/climate-disruptions.html). Additionally, only 17 percent of organizations regularly rehearse their response to major disruptions, and 25 percent have no mechanisms in place for building climate resilience.

Overall, respondents said their organizations have been impacted by the following:

- Power outages (32%)
- Supply chain disruptions (32%)
- Employee health (30%)
- Transportation disruptions (27%)
- Regulatory compliance issues (22%)

Power outages also topped the list of climate-related disruptions in Africa specifically (54% of respondents), with supply chain breakdowns (41%) and employee health issues (39%) leading in North America.

Despite this, two-thirds of total respondents say their organizations are not investing adequately to address the physical risks posed by climate change, and only 37 percent are planning to increase spending in this area.

“Organizations must make climate adaptation a priority — not only to safeguard their operations but to protect the people and places at risk,” said report author Emmeline Skelton, Head of Sustainability at ACCA. “Rising temperatures, more intense rainfall, and swelling sea levels make the evidence unmistakable: we must act now to build resilience and mitigate further harm.”

Read more at ACCA: https://www.accaglobal.com/uk/en/news/2024/November/climate-disruptions.html.

SUSE Cloud Observability Announced

SUSE has announced early access to SUSE Cloud Observability — a fully managed observability platform designed specifically for Rancher-managed Kubernetes clusters.

The platform lets enterprises “monitor mission-critical workloads in Rancher-managed Kubernetes clusters across AWS, Azure, and Google Cloud, quickly detecting and resolving issues in real time,” the announcement says (https://www.suse.com/news/SUSE-Eliminates-Need-for-Multiple-Monitoring-Tools/).

Benefits of SUSE Cloud Observability include:

- Rapid setup and deployment: Deploy quickly with a SaaS observability solution and pre-configured policies.
- Comprehensive insights: Deep insights into Kubernetes environments powered by OpenTelemetry, with more than 40 out-of-the-box dashboards providing a holistic view of the entire stack.
- Cost-effective entry point: Transparent, usage-based pricing with no hidden costs lets you start small and scale as needed.

To learn more, visit SUSE’s Cloud Observability Early Access Program: https://more.suse.com/SUSE_Cloud_Observability_Early_Access_Program.html.

Supercomputing Conference 2024

Bursting at the Seams

We'll recap some noticeable and not so noticeable points of interest that came up during the largest, most attended Supercomputing Conference yet. By Jeff Layton

Lead Image © djem, 123RF.com

SC24 took place in Atlanta, GA, November 17-22. As I'm writing this, 17,959 attendees – that's 3,000+ more than last year – registered. More than 500 companies filled the exhibition floor, which was a net new 136 companies over last year. Both attendance and corporate presence set records.

I can't possibly present a complete round-up or summary of SC24, but you can find other recaps – written, audio, and video – that cover a lot more detail than I can here. What I can do is present some things I found interesting. If you feel I've missed something important, please let me know. Perhaps you can write something about your thoughts.

One of my favorite things to do is attend the Beowulf Bash (BeoBash) [1]. It is the only open party during the conference, so a wide range of people attend. Alas, my stamina gave out this year, but about 1,300 people attended, which is also a new record. For an "open" party, the list of sponsors on the home page is amazing. The power of community is noticeable. Thanks Lara, Michael, and Doug. They, and others, are the unsung heroes putting this party together.

SCinet and Power

One thing I learned this year is that SCinet [2], which was used for Internet access during the conference, uses quite a bit of power. I don't know how SCinet started, but it demonstrates what it takes to build the most powerful and advanced network anywhere, including power and cooling, monitoring, fixing issues, and supporting users. A number of volunteers built and ran this global collaboration along with a respectable number of companies who donated hardware and allowed their employees to volunteer.

While attending SC24, I learned that SCinet uses 1MW/day. After a little Google search, I discoverd that's enough power for 1,000 American homes. This number led me to think about liquid-cooled networking. The power used by a single node today is quite high, forcing systems to be liquid cooled. Other options are possible but liquid is increasingly becoming the cooling technology for HPC servers. With large multirail systems, quite a bit of networking equipment is needed, especially for artificial intelligence (AI) and deep learning (DL) applications, where network operation is a key driver in performance. Higher clock speed network chips with an associated increase in power should result in the need for liquid cooling for networking equipment. What's going to make this "fun" is that you will probably have networking cables mixed with cooling tubes coming out of the front or back of the racks. How these will be organized will be interesting and probably the key to making them usable.

Overall, given the SCinet example of 1MW/day, I think liquid cooling of networking equipment is inevitable, and in the next year, you will see an experiment or two around this technology.

HPCG and Fugaku

The TOP500 project is built around a single test: High Performance Linpack (HPL). The High Performance Conjugate Gradient (HPCG), while still a single benchmark, has several components that stress different aspects of the system from HPL. The HPCG conjugate gradient code creates different measures of performance that go into the resulting HPCG performance [3] (reported in teraflops per second;

TFLOPS). Its design focuses on data access patterns of algorithms (e.g., sparse matrix computations) that are driven by memory bandwidth (irregular memory access) and the interconnect performance.

This year's number one for the HPCG Benchmark is Fugaku, which has been at the top of the list since June 2020. In the high-performance computing (HPC) world, being number one leads to the use of terms such as "dominant" when discussing the system. Fugaku was number one on the HPL list for a while and is number six today, but it has been number one on the HPCG list for four years. Let me illustrate why it is so dominant.

Fugaku was first on the June 2020 list at 13,400 TFLOPS. The previous system, Summit, achieved only 2,925.75 TFLOPS. Fugaku's results went up a little on the next list (November 2020), at almost 5.5 times faster than Summit, until November 2022 when Frontier became the number one system on the TOP500. However, Fugaku was still number one on the HPCG list, with Frontier close behind. Frontier achieved 14,054 TFLOPS, whereas Fugaku recorded an HPCG performance of 16,004.5 TFLOPS.

On the latest TOP500 list, November 2024, Frontier was number two, with El Capitan becoming number one, and Fugaku slipped to number six, but remained number one on the HPCG list; note that El Capitan did not appear on the HPCG list (perhaps yet).

Taking all this into account, especially considering the large increases in performance for the number one system on the TOP500, Fugaku remains a very strong HPC system and dominant in HPCG. Although I am not discussing the Green500, Fugaku is currently number 86 on the list and started at number nine in June 2020 when it joined the lists. A ranking of 86 sounds low, but this is a four-year-old system that is still doing amazing research. Although you can argue about the usefulness of HPCG as a system benchmark, when you see such great HPCG performance and very good HPL performance, it is difficult to fault Fugaku. Perhaps new systems aim at the HPL and not HPCG benchmark, which could explain why Fugaku is still number one on HPCG. However, these new systems also might not be as capable as Fugaku on the HPCG test. Inquiring minds want to know. I think it would be advantageous to study the design of Fugaku relative to current designs, because HPC and AI algorithms love memory bandwidth.

System Power and Cooling

It bears repeating that power and cooling is not a secondary consideration and is now a primary consideration, along with processing capability, memory, and networking. The exhibit floor at SC24 proved this beyond a shadow of a doubt. The show floor had a very large number of companies in this space, showing and discussing what they could do to help the HPC and AI industries reach their targets. These targets are a massive increase in computing power (resources) with the best possible energy efficiency and as little additional cost as possible.

Several companies showed a variety of liquid cooling options. In the past, companies offered liquid-cooled doors, but there has been a shift to direct cooling of the processors and perhaps other components in a server (e.g., memory). Typically, targeted components have heat transfer plates directly attached, through which a liquid runs to remove the heat. The heat is then transferred to a heat exchanger and dumped somewhere (usually outside the data center). Although it sounds simple, in practice it is not.

Several companies showed direct liquid-cooled server designs that are available now. The key thing to make sure of is that your total data center design, including not just servers but networking and the physical building, are ready for these servers. Therefore, power and cooling experts are a huge part of any new system discussion.

Other companies displayed immersive cooling. These solutions submerge the entire server without a case into a vat of liquid. This cooling technique has been around for a while. The Texas Advanced Computing Center (TACC) experimented with this method, and they called it the "chicken fryer" because the immersive fluid smelled like fried chicken. I think today's solutions have gone beyond this example, with several solutions being promoted on the floor.

I have to mention one vendor whose name shocked many people: Valvoline. They've been dealing with automotive oil and cooling for many years, including high-performance racing. Now they are taking their knowledge to data center cooling, which tells you how important cooling has become. They also won the most annoying marketing award for the most email received before I blocked them.

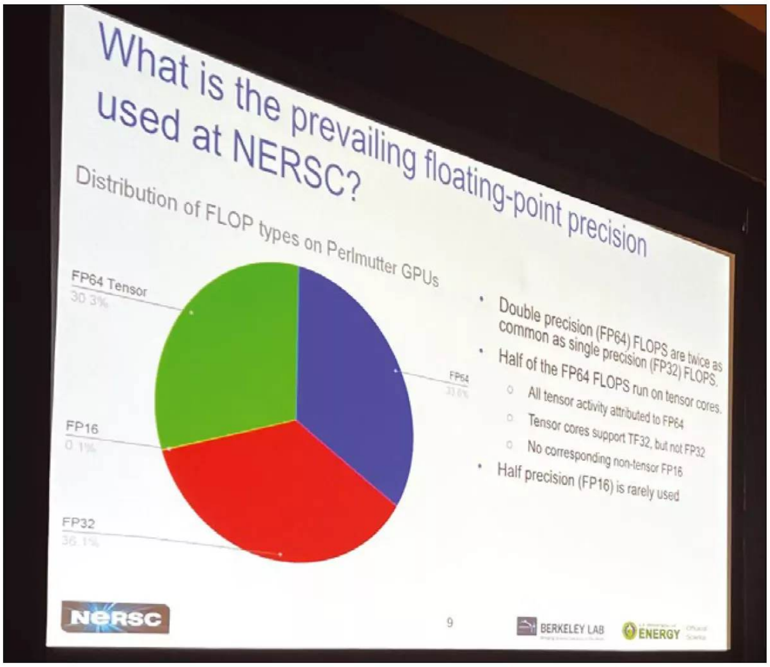
Reduced Precision

On Bluesky, the new social media hangout for HPC, Si Hammond from the National Energy Research Scientific Computing Center (NERSC) posted an image (Figure 1) he took during a talk at the 15th International Workshop on Performance Modeling, Benchmarking, and Simulation (PMBS24) [4] titled "System-Wide Roofline Profiling – A Case Study on NERSC's Perlmutter Supercomputer." [5]

Notice that whereas FP64 and FP64 Tensor dominate the overall floating-point precision on Perlmutter GPUs, FP32 usage is about 36 percent of the total. This report surprises me because the broad domain of simulation-based HPC applications still seem to be heavily based on FP64. The presenter points out that about half of the FP64 floating-point precision is Tensor based, but no other precisions use Tensor Cores. Overall, I think this result shows that researchers are using lower precision, perhaps by itself or in combination with higher precision in portions of the application. Moreover, they are using Tensor Cores, which indicates they are taking advantage of the hardware (which they should).

Other Topics

Almost everyone registered the big topics from SC24, such as a new



What is the prevailing floating-point precision used at NERSC?

Distribution of FLOP types on Perlmutter GPUs

- Double precision (FP64) FLOPS are twice as common as single precision (FP32) FLOPS.
- Half of the FP64 FLOPS run on tensor cores.
  - All tensor activity attributed to FP64
  - Tensor cores support TF32, but not FP32
  - No corresponding non-tensor FP16
- Half precision (FP16) is rarely used

Figure 1: **FLOP types at NERSC (cropped image from PMBS24 presentation by Si Hammond; used with permission).**

number one on the TOP500, the rise of AI, the increasing importance of power and cooling, and so on, because a great deal has been written about them. I also want to look at topics I think are important that slip under the radar, such as Open OnDemand [6] and MATLAB. Open OnDemand is a rapidly growing HPC and AI tool with a web-based portal that allows users to access HPC resources, which means you can get a compute node in a cluster, get a desktop running on that node, and then run interactive applications such as Jupyter notebooks or MATLAB.

I started noticing Open OnDemand a few years ago and found it to be an amazing tool. It can be argued that using a cluster node with high-speed networking and storage for interactive applications is underutilizing the resource, but the counter argument is that sometimes users need a very performant single node for interactive work, and creating a heterogeneous cluster to accommodate them leads to complications and a reduced number of high-performance compute nodes. Moreover, if a user needs a single high-performance node, a high-end workstation might not have enough compute resources, and they might not have a place to plug it in at their office or lab (too much power).

Caution is still needed when, for example, a user gets an entire node with eight GPUs, lots of CPU cores, terabytes of memory, and lots of local solid-state drive (SSD) storage and then just uses one GPU with maybe 16 cores (or worse, one core). This configuration might be important to the user, but underutilizing the resources in this case might be too much. Of course, you could allow multiple users to use the same node, so resources don't go to waste. The moral is, carefully consider the use case because it is becoming very common.

One of the more recent presentations I've seen on Open OnDemand, had the presenter explaining how they used their Tesla parked in their home driveway for a remote login to the cluster and then use Open OnDemand to get an interactive desktop on a compute node that displayed on the screen in their Tesla. Of course, it was a stunt, but it showed that Open OnDemand is very flexible and easy to use once configured.

Mike Croucher of MathWorks posted during SC24 that many HPC sites are using MATLAB on Open OnDemand. MATLAB is an extraordinarily popular tool in science and engineering. On Bluesky, Croucher posted a picture of MATLAB running on a University of Utah cluster [7]. I've run Open OnDemand on my small home lab clusters and it works just great. It's particularly popular with MATLAB users, because they can get access to a very powerful single node when their laptop or desktop doesn't have the resources they need.

The combination of MATLAB, one of the most powerful science and engineering tools, along with single servers with many GPUs (more than four) per server, terabytes of memory, and lots of local SSD storage that is all made available by Open OnDemand can enable researchers to go to the next level beyond a small laptop or even a desktop. You really can't take such a server and plug it in where a user sits or even in a lab. With Open OnDemand and powerful tools such as MATLAB and other easy-to-use tools, you can use multiple CPUs and GPUs and move up the food chain without being condemned to the end of the "long tail of science."[8] ■

Info

[1] Beowulf Bash: [https://beowulfbash.com/]

[2] SCinet: [https://sc24.supercomputing.org/scinet/]

[3] HPCG: [https://www.hpcg-benchmark.org/]

[4] PMBS24: [https://pmbs-workshop.github.io/]

[5] Austin, B., D. Kulkarni, B. Cook, S. Williams, and N.J. Wright. System-Wide Roofline Profiling – A Case Study on NERSC's Perlmutter Supercomputer. IEEE, 2024: DOI 10.1109/SCW63240.2024.00180: [https://conferences.computer.org/sc-wpub/pdfs/SC-W2024-6oZmigAQfgJ1GhPLOyE3pS/555400b391/555400b391.pdf]

[6] Open OnDemand: [https://openondemand.org/]

[7] Open OnDemand with MATLAB: [https://bsky.app/profile/walkingrandomly.bsky.social/post/3lbd6dxomtc26]

[8] Long tail of science: [https://voices.uchicago.edu/compinst/blog/unwinding-long-tail-science/]

Web-based network monitoring

In Sight

Bloonix integrates all the tools you need to monitor large IT infrastructures 24x7 to prevent failures. By Erik Bärwaldt

Permanent monitoring of IT infrastructures is a routine task for every network admin. Depending on the scope and complexity of existing installations, several tools are used for this purpose. Bloonix is a modular monitoring tool that combines numerous services in a single interface, offering a quick overview of network components.

The Idea

Bloonix [1], written in JavaScript and Python 3, is a server-based application for monitoring complex, distributed IT infrastructures. The tool comprises five components: the Bloonix server, a webGUI for display purposes, field information from agents that need to be installed separately, plugins and a satellite. Satellites must be configured separately and enable the service to be checked from remote locations. You can install agents on the servers to be monitored or set them up on a central machine that documents the function of network components such as a router or switch.

Bloonix uses several protocols for monitoring devices, (e.g., the Simple Network Management Protocol, which are supported by plugins). The server itself offers a graphical interface to visualize the status of each system with the help of the WebGUI. The server stores the status data for the IT components in a database. If critical events occur, Bloonix can sound the alert and notify the responsible administrator by email or text message.

The agents log on to the server and automatically transmit their data after the initial configuration, without the need for the server to pull. This method translates to lower configuration overhead because you do not have to set up different port shares for the server to enable client access.

The modular structure of the application allows for very flexible use. Plugins monitor Linux and Windows systems, web servers such as Apache or NGINX, and databases such as PostgreSQL or MySQL. Nagios plugins are also compatible with Bloonix.

The Interface

You can host Bloonix yourself. All you need is a Linux system with a web browser on the client side supporting JavaScript. The Bloonix server does not need specific distributions, web servers, or databases. Installation instructions can be found on the project website, which is still pretty much unfinished if you consider the new versions for Debian [2] and Rocky Linux [3].

After the basic installation, use systemd to start the server, the web interface, and the agents. You can then call up the Bloonix web interface in the web browser on *http://127.0.0.1:9000* and log in as *admin* with the default password *bloonix*.

The service's web interface first opens a dashboard on which you can view the active services and check out the event display (Figure 1), which lists the active services and events in a table that Bloonix updates at short intervals. In the Status column, you can immediately see by the color blocks whether the services are working properly and whether important events require manual intervention.

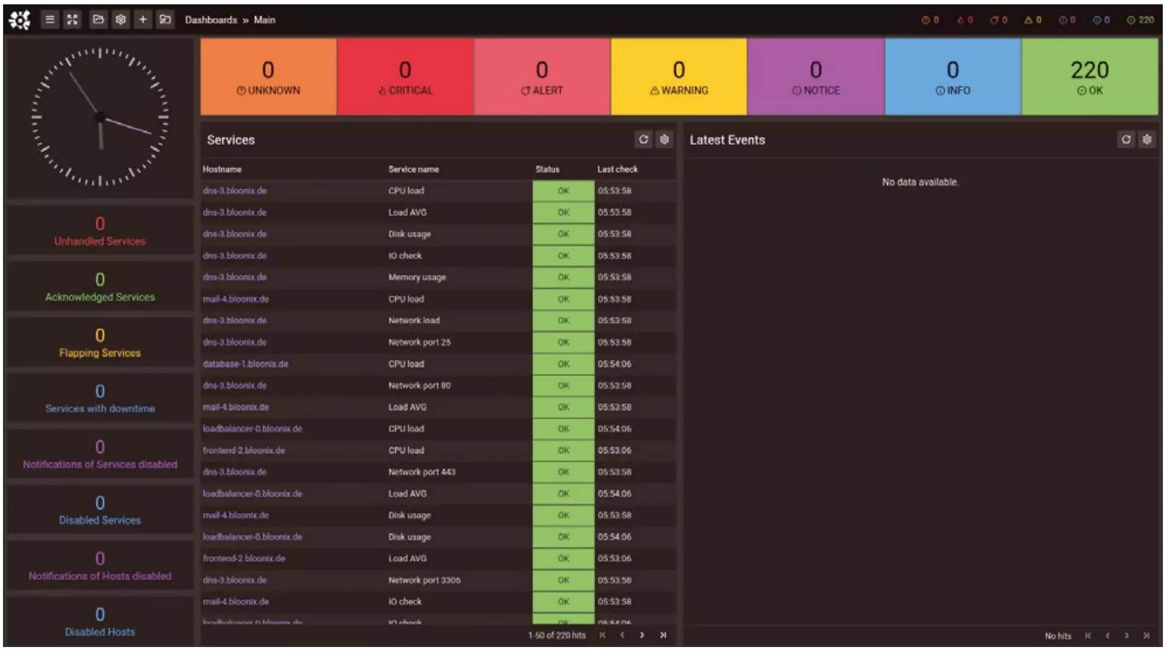
The individual displays in the web interface can be customized to a large extent. You can start by configuring some basic settings from the hamburger menu displayed at top left in the small icon menubar. The *My Settings* category lets you adjust the language, time zone, password, and appearance in the respective dialogs. Status messages appear in a tile bar

Lead Image © arasdubov, 123RF.com

at the top of the window. Bloonix also displays them in more detail in a column on the left.

Plugins

Bloonix can monitor a wide variety of components with the help of plugins. Besides information transmitted by SNMP, such as CPU performance, memory requirements, mass storage utilization, and network-based categories such as link status, the tool also directly queries numerous system states of Linux installations, which, in addition to hardware-specific queries, include filesystem statuses. Another plugin monitors the usage of network protocols and their statuses. Bloonix supports application-specific protocols such as IMAP, SMTP, HTTP, and POP3 and their secure counterparts. Bloonix can also monitor server-based services such as web servers and databases for consistency.



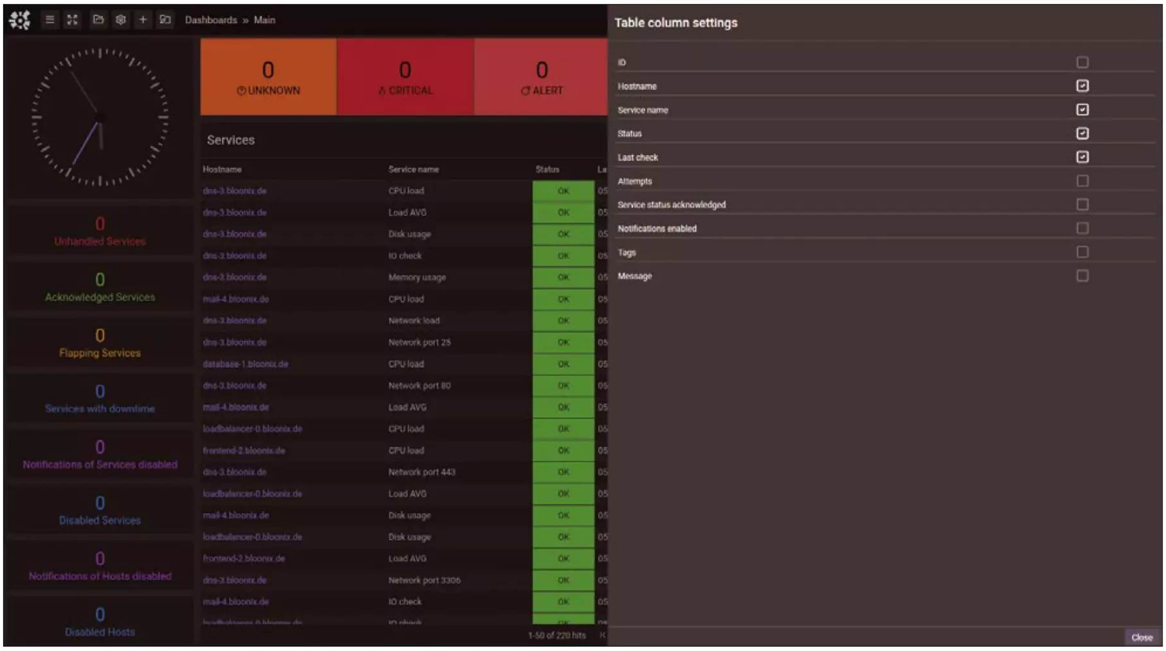
Dashboards » Main
0 UNKNOWN
0 CRITICAL
0 ALERT
0 WARNING
0 NOTICE
0 INFO
220 OK
0 Unhandled Services
0 Acknowledged Services
0 Flapping Services
0 Services with downtime
0 Notifications of Services disabled
0 Disabled Services
0 Notifications of Hosts disabled
0 Disabled Hosts
Latest Events
No data available.
No hits

Services

Hostname	Service name	Status	Last check
dns-3.bloonix.de	CPU load	OK	05:53:58
dns-3.bloonix.de	Load AVG	OK	05:53:58
dns-3.bloonix.de	Disk usage	OK	05:53:58
dns-3.bloonix.de	IO check	OK	05:53:58
dns-3.bloonix.de	Memory usage	OK	05:53:58
mail-4.bloonix.de	CPU load	OK	05:53:58
dns-3.bloonix.de	Network load	OK	05:53:58
dns-3.bloonix.de	Network port 25	OK	05:53:58
database-1.bloonix.de	CPU load	OK	05:54:06
dns-3.bloonix.de	Network port 80	OK	05:53:58
mail-4.bloonix.de	Load AVG	OK	05:53:58
loadbalancer-0.bloonix.de	CPU load	OK	05:54:06
frontend-2.bloonix.de	CPU load	OK	05:53:06
dns-3.bloonix.de	Network port 443	OK	05:53:58
loadbalancer-0.bloonix.de	Load AVG	OK	05:54:06
mail-4.bloonix.de	Disk usage	OK	05:53:58
loadbalancer-0.bloonix.de	Disk usage	OK	05:54:06
frontend-2.bloonix.de	Load AVG	OK	05:53:06
dns-3.bloonix.de	Network port 3306	OK	05:53:58
mail-4.bloonix.de	IO check	OK	05:53:58

1-50 of 220 hits

Figure 1: Bloonix initially shows the monitored components in a table.



Dashboards » Main
Services
Close

Table column settings

☐ ID
☑ Hostname
☑ Service name
☑ Status
☑ Last check
☐ Attempts
☐ Service status acknowledged
☐ Notifications enabled
☐ Tags
☐ Message

Figure 2: Customize your table views in a clear-cut dialog.

Control

The overview tables can be customized, if needed, from the gear icon at top right in each table. Clicking it opens an options menu where you can display or hide various columns by checking or unchecking the boxes (Figure 2). Your changes take effect immediately after pressing *Close* to close the dialog; you do not need to save the changes separately.

From the hamburger menu at top left, you can display all the system statuses in detail and in individual categories. To do so, use the *Hosts*, *Services*, and *Events* options. The display then switches to a table view with additional information shown for each entry. Among other things, you can check the utilization status of mass storage devices and processors and the current memory requirements.

Checking the box to the left of a table row lets you set various options for the service or host. In this way, you can configure and schedule maintenance work in advance or switch services on or off. You can also modify the monitoring intervals for the services by pressing the button at bottom left in the window after selecting the services.

To implement a single action for several services, select all the processes and hosts in which you are interested. In the bar at bottom left, a field shows the number of selected services or hosts. This view is good in terms of clarity, especially for large-scale installations with massive tables. As soon as you select an action with one of the buttons to the right, Bloonix executes the desired function simultaneously against all the selected table rows.

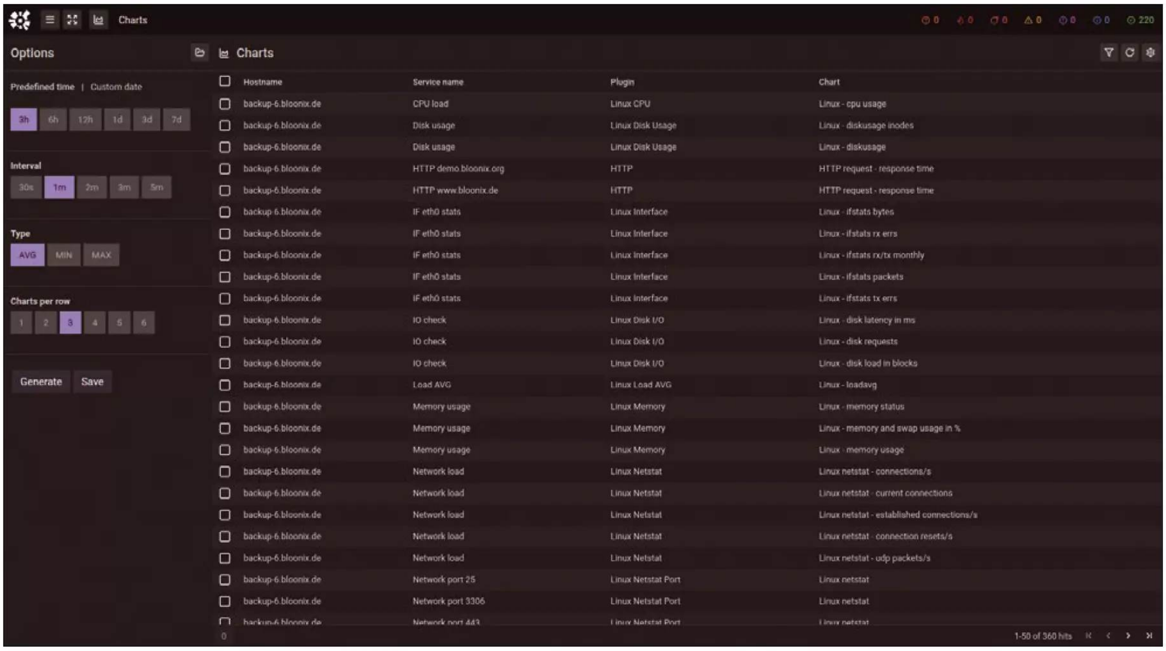
Modular

Because of the large amount of information provided, it is easy to lose track, especially in large-scale distributed IT infrastructures with hundreds of table entries. Additionally, some services hardly require any maintenance, which makes prioritization of the dashboard view advisable. Thanks to the individual tables consisting of dashlets (i.e., elements you can modify), you can customize the dashboard to suit your needs quite easily.

Bloonix offers a large number, of wide-ranging design options, which you can enable by pressing the *Edit dashboard* button in the bar at top left in the window. Bloonix then highlights all the elements on the dashboard with dashed frames. Additionally, up to four buttons for editing the display appear above each element. The *Remove the dashlet* buttons ("x") above each frame hide the respective dashlet completely. The space this frees up can then be used for other dashlets. You can use the button with the stopwatch symbol (*Reload interval*) to adjust the interval for refreshing the display for each dashlet. When you press this button, Bloonix opens a dialog with eight fields to *disable* or select intervals between 10 seconds and 10 minutes.

You can also completely disable the display refresh separately for each dashlet, which means you can update the display for less important services less frequently and monitor services or hosts with greater relevance more closely and at shorter intervals.

To modify the way the field for each service is displayed, use the gear button (*Configure*) to define new background and text colors or modify the title displayed for the respective process. Because these labels are free text entries, they do not change if you change the language settings. If you want to use a different language, you have to modify



☐	Hostname	Service name	Plugin	Chart
☐	backup-6.bloonix.de	CPU load	Linux CPU	Linux - cpu usage
☐	backup-6.bloonix.de	Disk usage	Linux Disk Usage	Linux - diskusage inodes
☐	backup-6.bloonix.de	Disk usage	Linux Disk Usage	Linux - diskusage
☐	backup-6.bloonix.de	HTTP demo.bloonix.org	HTTP	HTTP request - response time
☐	backup-6.bloonix.de	HTTP www.bloonix.de	HTTP	HTTP request - response time
☐	backup-6.bloonix.de	IF eth0 stats	Linux Interface	Linux - ifstats bytes
☐	backup-6.bloonix.de	IF eth0 stats	Linux Interface	Linux - ifstats rx errs
☐	backup-6.bloonix.de	IF eth0 stats	Linux Interface	Linux - ifstats rx/tx monthly
☐	backup-6.bloonix.de	IF eth0 stats	Linux Interface	Linux - ifstats packets
☐	backup-6.bloonix.de	IF eth0 stats	Linux Interface	Linux - ifstats tx errs
☐	backup-6.bloonix.de	IO check	Linux Disk I/O	Linux - disk latency in ms
☐	backup-6.bloonix.de	IO check	Linux Disk I/O	Linux - disk requests
☐	backup-6.bloonix.de	IO check	Linux Disk I/O	Linux - disk load in blocks
☐	backup-6.bloonix.de	Load AVG	Linux Load AVG	Linux - loadavg
☐	backup-6.bloonix.de	Memory usage	Linux Memory	Linux - memory status
☐	backup-6.bloonix.de	Memory usage	Linux Memory	Linux - memory and swap usage in %
☐	backup-6.bloonix.de	Memory usage	Linux Memory	Linux - memory usage
☐	backup-6.bloonix.de	Network load	Linux Netstat	Linux netstat - connections/s
☐	backup-6.bloonix.de	Network load	Linux Netstat	Linux netstat - current connections
☐	backup-6.bloonix.de	Network load	Linux Netstat	Linux netstat - established connections/s
☐	backup-6.bloonix.de	Network load	Linux Netstat	Linux netstat - connection resets/s
☐	backup-6.bloonix.de	Network load	Linux Netstat	Linux netstat - udp packets/s
☐	backup-6.bloonix.de	Network port 25	Linux Netstat Port	Linux netstat
☐	backup-6.bloonix.de	Network port 3306	Linux Netstat Port	Linux netstat
☐	backup-6.bloonix.de	Network port 443	Linux Netstat Port	Linux netstat

Figure 3: Charts can be generated quickly from lists of services.

the labels again in the edit dialog. All changes made in this area need to be saved manually. You can do this by clicking the *Submit* button at bottom right in the settings dialog.

As soon as you hover over one of the dashlets, Bloonix displays double arrows in the corners of the dashlet frame, so you can adjust the frame size by dragging the mouse while holding the left mouse button. Reducing the size of less relevant displays gives you extra space for additional dashlets.

Once you have configured all the settings and customized the dashboard to your liking, it is a good idea to save the new settings by using the *Save dashboard* button at top left in the horizontal buttonbar so that Bloonix loads them again the next time you log out.

Charts

A picture is worth a thousand words, and graphics really do give you a far better overview of what is happening on the network than endless columns of numbers. History charts let you record a system status far faster than in numerical tables. To do so, you can select particularly important values in Bloonix and display them simultaneously on the screen in the form of history charts.

The first step is to open the *Charts* option in the hamburger menu at top left. A table with the active hosts and the services running on them then appear (Figure 3). Next, select the services intended for the graphical display by checking the boxes to the left of their entries. You can use the Options section on the left to define further details, such as the number of charts to be displayed in each line and the refresh intervals for the charts. These options apply to all charts.

Clicking *Generate* tells Bloonix to switch to a graphical display of all selected table entries (Figure 4), which then output data according to the specified options. You can obtain detailed information on each table entry by hovering over the graph in question. The tool displays the date and time of the entry as you scrub over the *X*-axis and shows further details in a separate small window.

Click *Charts* in the top left of the small toolbar to return to the table view; you can make changes or select additional table entries here for the graphical history display. After returning to the dashboard, you can call up the chart display again, but this will take you back to the table view configuration dialog.

To transfer charts permanently to a dashboard, switch to the dashboard display. When you get there, press *Edit dashboard* at top left, and then click on the buttons to remove from the display the dashlets you do not need. When done, press *Add a dashlet* in the same bar. In the dialog that now appears, select one of the listed dashlets and click on the *Chart* option for graphs.

The software now opens the dialog for generating charts, and you can define the status displays you need. After a final click on *Generate*, the application switches to the dashboard and shows the graphical displays as dashlets in free spaces (Figure 5). Like all other segments of the dashboard, these can be edited manually. Finally, click *Save dashboard* to save the new layout so that it is displayed again when you log back in.

In the same way, you can design individual dashboards in larger IT infrastructures by selecting the *Create a new dashboard* option (" + ") on the primary dashboard and adding the

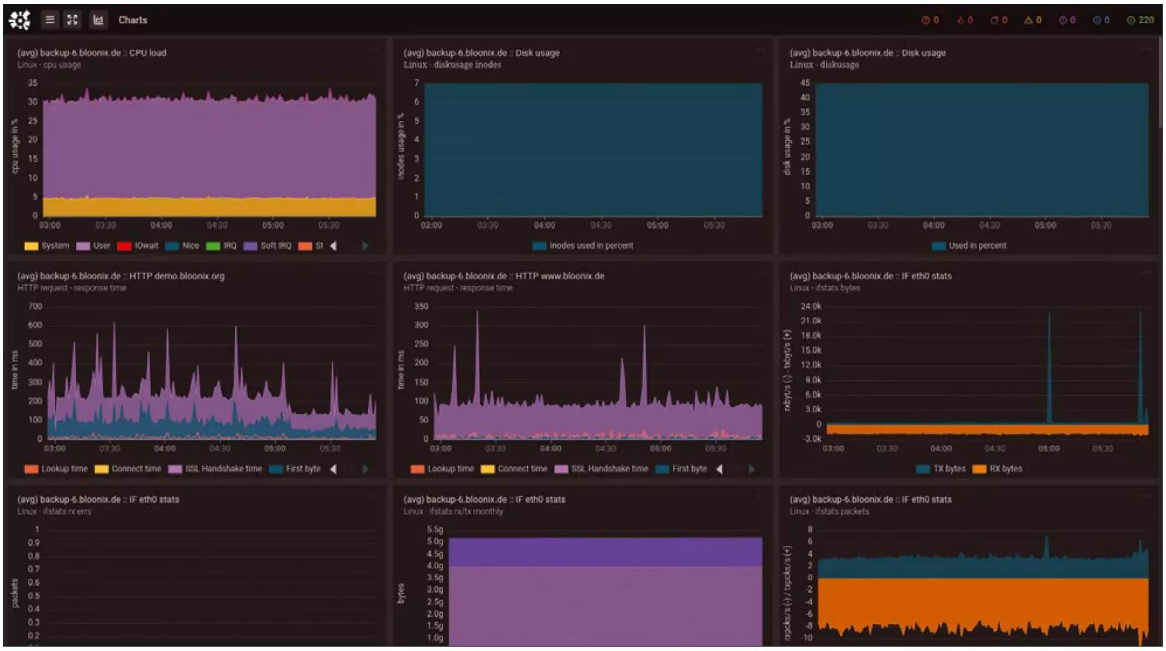

Charts
0 0 0 0 0 0 220
(avg) backup-6.bloonix.de :: CPU load
Linux - cpu usage
cpu usage in %
35 30 25 20 15 10 5 0
03:00 03:30 04:00 04:30 05:00 05:30
System User IOwait Nice IRQ Soft IRQ St
(avg) backup-6.bloonix.de :: Disk usage
Linux - diskusage inodes
inodes usage in %
7 6 5 4 3 2 1 0
Inodes used in percent
(avg) backup-6.bloonix.de :: Disk usage
Linux - diskusage
disk usage in %
45 40 35 30 25 20 15 10 5 0
Used in percent
(avg) backup-6.bloonix.de :: HTTP demo.bloonix.org
HTTP request - response time
time in ms
700 600 500 400 300 200 100 0
Lookup time Connect time SSL Handshake time First byte
(avg) backup-6.bloonix.de :: HTTP www.bloonix.de
HTTP request - response time
time in ms
350 300 250 200 150 100 50 0
Lookup time Connect time SSL Handshake time First byte
(avg) backup-6.bloonix.de :: IF eth0 stats
Linux - ifstats bytes
rxbyt/s (-) - txbyt/s (+)
24.0k 21.0k 18.0k 15.0k 12.0k 9.0k 6.0k 3.0k 0 -3.0k
TX bytes RX bytes
(avg) backup-6.bloonix.de :: IF eth0 stats
Linux - ifstats rx errs
packets
1 0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2
(avg) backup-6.bloonix.de :: IF eth0 stats
Linux - ifstats rx/tx monthly
bytes
5.5g 5.0g 4.5g 4.0g 3.5g 3.0g 2.5g 2.0g 1.5g 1.0g
(avg) backup-6.bloonix.de :: IF eth0 stats
Linux - ifstats packets
rxpcks/s (-) / txpcks/s (+)
8 6 4 2 0 -2 -4 -6 -8 -10

Figure 4: The graphical display provides a quick overview of important system statuses.

desired dashlets to the new overview after assigning them names. In this way, you can design tailored displays for specific hosts and processes and always keep track of the critical components.

Conclusions

Bloonix is a powerful and modern monitoring tool for extensive IT infrastructures that lets you monitor individual hosts and services in a convenient and clear way. On the downside, the documentation for the software currently leaves a lot to be desired, and the information on the Bloonix project page is incomplete. Third-party sources on the Internet almost exclusively refer to a totally obsolete version of the tool, which also had a different graphical user interface. If the developers of the tool clear up these deficits, Bloonix will certainly be able to compete with rivals such as Zabbix or Nagios. ■

Info

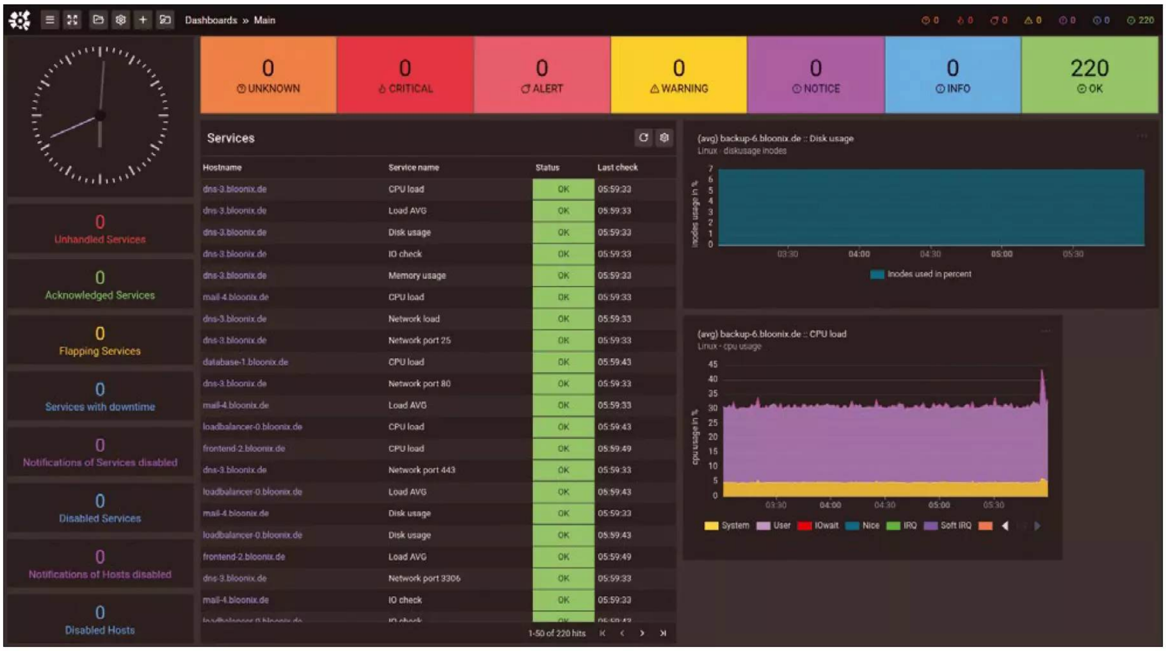
[1] Bloonix: [https://www.bloonix.org]

[2] Documentation for Debian: [https://docs.bloonix.org/en/installation/debian]

[3] Documentation for Rocky Linux: [https://docs.bloonix.org/en/installation/rockylinux]

Author

Erik Bärwaldt is a self-employed IT admin and technical author living in United Kingdom. He writes for several IT magazines.


Dashboards » Main
0 UNKNOWN
0 CRITICAL
0 ALERT
0 WARNING
0 NOTICE
0 INFO
220 OK
0 Unhandled Services
0 Acknowledged Services
0 Flapping Services
0 Services with downtime
0 Notifications of Services disabled
0 Disabled Services
0 Notifications of Hosts disabled
0 Disabled Hosts
(avg) backup-6.bloonix.de :: Disk usage
Linux - diskusage inodes
inodes usage in %
03:30
04:00
04:30
05:00
05:30
Inodes used in percent
(avg) backup-6.bloonix.de :: CPU load
Linux - cpu usage
cpu usage in %
System
User
IOwait
Nice
IRQ
Soft IRQ

Services

Hostname	Service name	Status	Last check
dns-3.bloonix.de	CPU load	OK	05:59:33
dns-3.bloonix.de	Load AVG	OK	05:59:33
dns-3.bloonix.de	Disk usage	OK	05:59:33
dns-3.bloonix.de	IO check	OK	05:59:33
dns-3.bloonix.de	Memory usage	OK	05:59:33
mail-4.bloonix.de	CPU load	OK	05:59:33
dns-3.bloonix.de	Network load	OK	05:59:33
dns-3.bloonix.de	Network port 25	OK	05:59:33
database-1.bloonix.de	CPU load	OK	05:59:43
dns-3.bloonix.de	Network port 80	OK	05:59:33
mail-4.bloonix.de	Load AVG	OK	05:59:33
loadbalancer-0.bloonix.de	CPU load	OK	05:59:43
frontend-2.bloonix.de	CPU load	OK	05:59:49
dns-3.bloonix.de	Network port 443	OK	05:59:33
loadbalancer-0.bloonix.de	Load AVG	OK	05:59:43
mail-4.bloonix.de	Disk usage	OK	05:59:33
loadbalancer-0.bloonix.de	Disk usage	OK	05:59:43
frontend-2.bloonix.de	Load AVG	OK	05:59:49
dns-3.bloonix.de	Network port 3306	OK	05:59:33
mail-4.bloonix.de	IO check	OK	05:59:33

1-50 of 220 hits

Figure 5: History graphs can also be displayed in the dashboard.

■

CLOUDFEST

March 17-20, 2025
Europa-Park, Germany

The world's largest cloud industry event is ready to once again take over a spectacular European amusement park to facilitate new partnerships, deep knowledge sharing, and the best parties the industry has ever seen.

8,700+ Participants
150+ Partners
250+ Speakers
80 Countries

Start your CloudFest Journey
AND SAVE €499!
With your FREE Code: **CF25Admin**

scan me!

reg.cloudfest.com

cloudfest.com

Monitor applications end to end

From Start to Finish

Your web server is running perfectly according to Nagios, but is it delivering the intended user experience? End-to-end monitoring lets you know whether your application is performing as expected. By Tim Schürmann

All of your systems seem to be working normally, but virtually nobody has placed an order in the online store since yesterday afternoon. A painstaking manual investigation reveals that the shopping cart takes an agonizingly long time to redirect to checkout – the problem is caused by a messed up cache. Nasty bugs like this can quickly cause expensive damage, but they often slip through the fingers of classic monitoring solutions because they only check whether the web server and PHP are running, not whether the web application being served to your customers is still working correctly and, above all, smoothly. For this problem, you need a special monitoring system. The right solution will check all of your real users as they fill the shopping cart and check out. Alternatively, or additionally, it will slip into the role of a real customer and call up critical functions on a regular basis. For example, the monitoring system could run through a complete ordering process in the online store or register as an author in a blog. In each case, the tool then records the time it takes to complete the action. In this way, you find out whether your responsive design for smartphones is accidentally concealing important controls and whether logging into the online store is suddenly taking several minutes to complete.

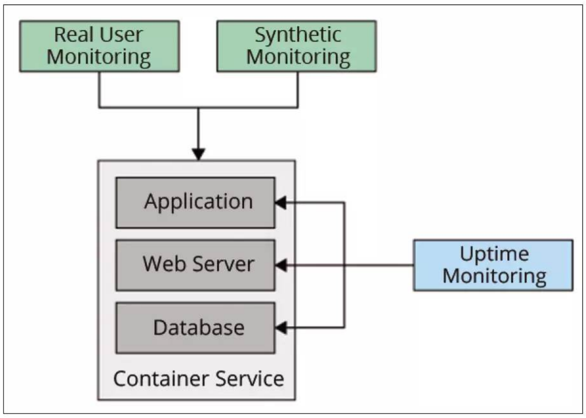


Figure 1: E2E monitoring observes the users (real user monitoring) and regularly simulates typical user actions (synthetic monitoring). The information and performance values obtained in this way supplement conventional uptime monitoring.

In contrast to traditional infrastructure or system monitoring, these tools continuously monitor software quality from the user's perspective. The monitoring tools automatically check all of the components involved in the background, from the database to the user interface at the other end. This process is called end-to-end (E2E) monitoring (Figure 1) and is primarily used for web applications (see the "Confusion of Terms" box). However, it can also be used for native Linux programs and smartphone apps. E2E monitoring is even suitable for applications that do not have a graphical user interface. If the services are accessible by REST API, for example, a monitoring solution would use those endpoints to do its work.

Timed Out

If the shopping cart responds very slowly, annoyed customers are likely to cancel the order process eventually. Administrators, on the other hand, want to know whether the software in their containers and on their

Photo by Adam Winger on Unsplash

virtual machines is currently working at its limits. More than just banks have to adhere to mandatory transaction times for their customers, which is why the speeds and response times determined by E2E monitoring are considered to be particularly valuable information. The performance data obtained in this way gives the development department the opportunity to hone the response behavior of the shopping cart in a targeted way. Administrators can book the computing power that is required and optimize capacity planning by doing so. They can also quickly find out whether the subscribed cloud services or software-as-a-service (SaaS) services are slowing down their applications.

Confusion of Terms

Although end-to-end monitoring is a common term, the strategy is frequently referred to by other names, such as end-user monitoring (EUM), end-user experience monitoring (EUXM), app monitoring, application performance management (APM), and chain monitoring.

In each case, though, the terms only refer to one aspect of E2E monitoring. Whereas APM exclusively focuses on speed measurements down to the code level, EUM checks how well and quickly the user interface performs from the user's perspective. Both EUM and the real user monitoring mentioned in the article can therefore be part of APM at the same time. Application response time monitoring, which measures the response time of an application, is related. Chain monitoring observes a system consisting of several components, such as a LAMP stack (Linux, Apache, MySQL, Python [Perl, or PHP]) that delivers WordPress.

End-to-end-monitoring combines approaches: An E2E monitoring solution measures the performance and quality of an entire application from the user's point of view, or at the point of customer contact if you prefer. Digital experience monitoring (DEM) is sometimes used as a synonym. On the Internet and in literature, however, all of these terms are often confused and used in a vague manner. Marketing departments do the rest to perfect this confusion of terms.

An E2E monitoring solution periodically monitors the application and logs the response times determined in the process. The reports produced in this way can be used to demonstrate compliance with a service level agreement (SLA) to a customer. At the same time, the logs let admins peek into the future. For example, if the shopping cart performance continuously slows down after 3:00pm, suitable extrapolation methods can be used to determine when the shopping cart will no longer respond and allow developers and administrators to take action in good time and prevent the worst case from happening. In other words, E2E monitoring identifies weak points at an early stage, even in complex components, and helps to prevent downtime.

Real User Monitoring

To obtain the performance data, E2E monitoring observes the real users as they progress through the ordering process. The monitoring tool measures the speed of the individual actions and records any problems that occur. This approach is known as real user monitoring (RUM). In this passive monitoring approach, the software only keeps an eye on the processes and does not take any action itself.

Ideally, measurement data is acquired in real time, so you need to choose carefully the tool you deploy to keep an eye on critical services. RUM proves to be extremely helpful when analyzing marketing campaigns. From the measured values, you can see precisely what is triggering the amazing discount on a particular product, where backlogs are suddenly appearing in the online store, and how customers are responding to offers. The data can then be used to prepare the online store for future campaigns to meet your expectations.

Enjoy – But Go Easy

Because of the way it works, RUM will tend to generate large volumes of data extremely quickly. Although it allows for detailed analysis of the problems and performance, the evaluation process itself is anything but trivial. As a rule, you will need to define measuring points at important places in the application to avoid the need for the software to record every mouse click in the shopping cart, so it can focus, for example, on the time taken from the cart being called up through the payment process. Wherever you have large numbers of microservices operating in the background, a huge amount of unimportant information can make its way into the measurement data. This noise distorts the analyses. The right approach is to reduce the noise by cleverly selecting measurement points and other indicators. Depending on the infrastructure and application, setting up RUM can be relatively time consuming – and you will have other problems, too.

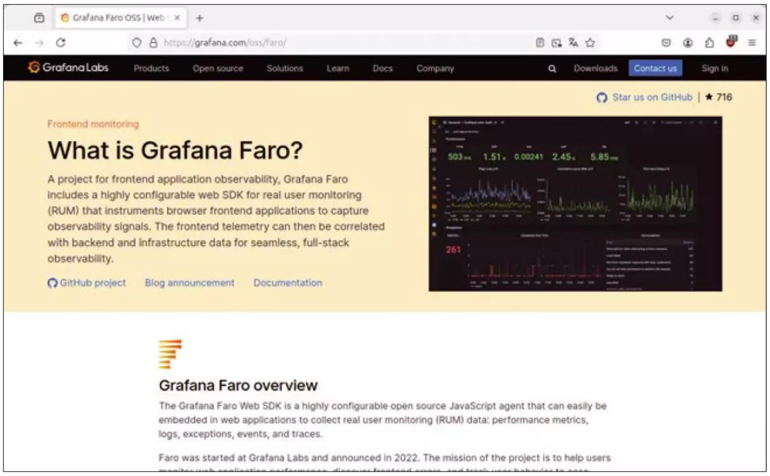
Real user monitoring tracks users through the application, which inevitably raises a data protection issue. Monitoring needs to be anonymized. Additionally, RUM does not cover all possible actions and functions of an application. For example, you have to wait until a user pays by credit card if you want to cover potential performance problems with the payment service provider. What's more, RUM only discovers a problem after the milk has been spilled and the potential customer is faced with a shopping cart that no longer works.

Integration Issues

Real user monitoring also somehow needs to know when a user triggers a relevant action, which is only possible if the application is set up in the right way or if suitable monitoring functions can be integrated directly into the source code at appropriate points. Monitoring solution developers often provide a software development kit (SDK) for this purpose. In the case of web applications, measurements must be made in the user's browser, which is usually carried out by a JavaScript snippet (Figure 2). However, ad blockers and other protections can prevent or even falsify the measurements.

Very few RUM tools on the market have an open source license. Grafana Faro is particularly popular, although it focuses on web applications **[1]**. Basic RUM **[2]** (**Figure 3**) is an alternative. For native Linux applications in particular, you should look into implementing telemetry data collection functions yourself or integrating suitable telemetry data libraries. After all, you know your application best and can forward the measured values to a monitoring back end that might already be running, such as the Grafana stack. For Go programs, for example, the OpenTelemetry-Go library **[3]** collects performance and other metrics according to the OpenTelemetry standard and sends them to compatible analysis platforms.

When monitoring third-party software, the source code is rarely available. In this case, the monitoring application needs to analyze the user interface displayed on the user's screen, much like spyware, and use optical character recognition (OCR), for example, to identify text content. The process is flawed and does not provide exact performance values, but you can work around the issue in quite an elegant way.


Grafana Faro OSS | Web
https://grafana.com/oss/faro/
Grafana Labs
Products
Open source
Solutions
Learn
Docs
Company
Downloads
Contact us
Sign in
Star us on GitHub | ★ 716
503 ms
1.51 s
0.00241
2.45 s
5.85 ms
261

Frontend monitoring

What is Grafana Faro?

A project for frontend application observability, Grafana Faro includes a highly configurable web SDK for real user monitoring (RUM) that instruments browser frontend applications to capture observability signals. The frontend telemetry can then be correlated with backend and infrastructure data for seamless, full-stack observability.

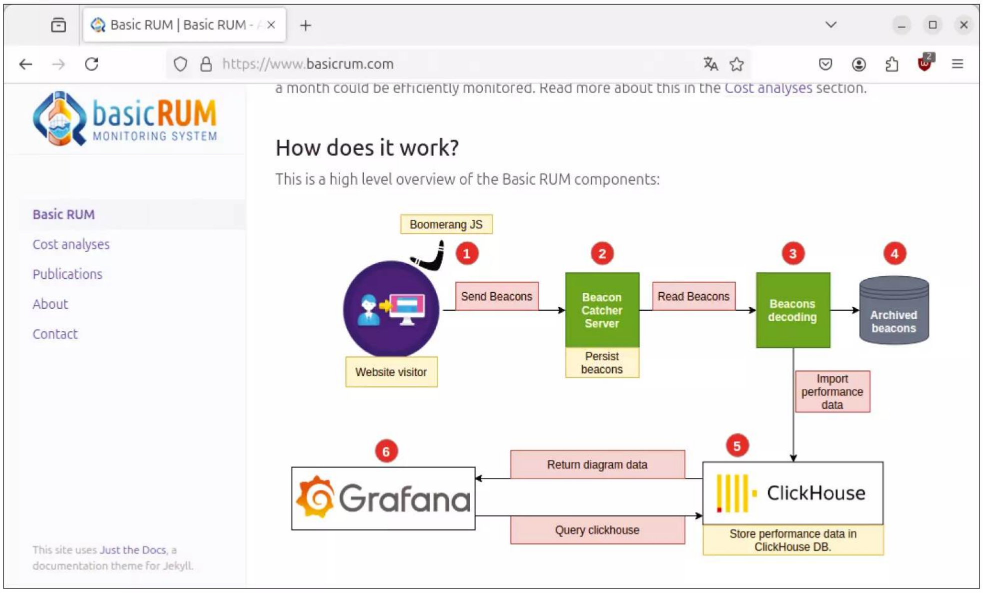
GitHub project Blog announcement Documentation

Grafana Faro overview

The Grafana Faro Web SDK is a highly configurable open source JavaScript agent that can easily be embedded in web applications to collect real user monitoring (RUM) data: performance metrics, logs, exceptions, events, and traces.

Faro was started at Grafana Labs and announced in 2022. The mission of the project is to help users

Figure 2: Grafana Faro consists of a JavaScript agent that you integrate into your web application and that stores the monitored data on a Grafana stack.


Basic RUM | Basic RUM -
https://www.basicrum.com
basicRUM
MONITORING SYSTEM
Basic RUM
Cost analyses
Publications
About
Contact
Boomerang JS
1
Website visitor
Send Beacons
2
Beacon Catcher Server
Persist beacons
Read Beacons
3
Beacons decoding
4
Archived beacons
Import performance data
5
ClickHouse
Store performance data in ClickHouse DB.
6
Grafana
Return diagram data
Query clickhouse

a month could be efficiently monitored. Read more about this in the Cost analyses section.

How does it work?

This is a high level overview of the Basic RUM components:

This site uses Just the Docs, a documentation theme for Jekyll.

Figure 3: Basic RUM relies on some well-known components, such as BoomerangJS for data acquisition and Grafana for data storage.

Synthetic Monitoring

E2E monitoring can act like a real user and click through the application along a predefined path. For example, you could tell the monitoring tool to call up the online store in the browser, click on *Register* in the top right-hand corner, and enter a username and password in the two fields. Along with other such actions, the monitoring software simulates the actions you would expect visitors to take on the site.

This approach is known as synthetic monitoring. The term "synthetic" makes it clear that the monitoring software does not make real requests to the application, but predefined requests instead. Because the monitoring system is interacting with the application, it is considered an active monitoring approach. Synthetic monitoring always sees the application as a black box (i.e., it cannot access other parts of the user interface and, e.g., make changes directly in the database).

Play It Again

In synthetic monitoring, you know both the inputs and the matching outputs. For example, after logging into the online store, you will want the splash page to reappear. A suitable monitoring tool then can directly determine whether the behavior deviates from the desired behavior and report specific errors in the application. Additionally, several passes through the shopping cart can be compared, which is particularly useful for performance measurements, allowing you to determine quickly whether requests from provider A's network are answered faster than those from provider B's network. Programmers and administrators are likely to be experiencing déjà vu by now. In software development, the programs you create are tested by end-to-end tests in exactly the same way. In many organizations, tools also monitor users and automatically replay their actions as required. For example, an invoice received by email is immediately transferred to the document management system. Administrators are familiar with such helpers and refer to this as Robotic Process Automation (RPA).

Synthetic Tools

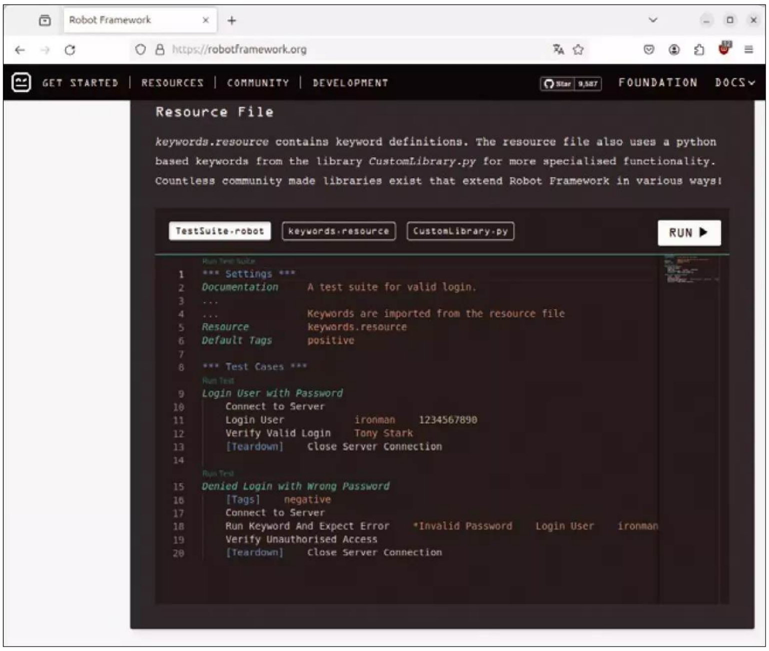
The boundaries to E2E monitoring are fluid: Many RPA and test tools are equally suitable for synthetic monitoring, as is the case with the Robot Framework [4], for example. The open source toolbox helps with test automation and RPA but has also proven its value in E2E monitoring (Figure 4). The framework can easily be extended to include custom features, and test cases can be written in your preferred in-house scripting language. Its syntax is now understood by Visual Studio Code and PyCharm; the Robot Framework also provides its own development environment, (RIDE) [5].

Robot Framework integrates the RobotMK extension [6] into the CheckMK monitoring system (Figure 5). CheckMK 2.3 is available in the form of the CheckMK Synthetic Monitoring [7] add-on. In all cases, the monitoring system ensures that the tests are always carried out automatically, and the CheckMK interface prepares the results. The advantage of integration is that you can continue to work with the monitoring system you are used to and do not have to maintain two completely different monitoring solutions. The Zabbix monitoring system can also carry out limited synthetic monitoring for web applications [8].

For web applications, you will also frequently see old acquaintances such as Cypress [9] and the Selenium Framework [10], which uses scripting to control the browser and the pages opened in it remotely. The test tool can record a user's mouse clicks and replay them later, which in turn significantly simplifies the process of creating hands-on tests (Figure 6).

A Question of the Mix

At the end of the day, synthetic monitoring only simulates typical user actions. Real users can behave in completely different ways. In practice,



Resource File

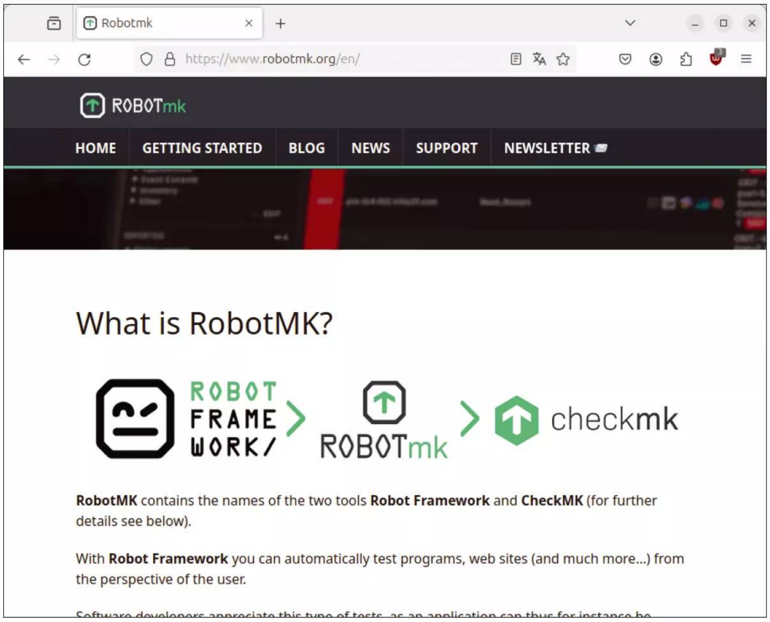
keywords.resource contains keyword definitions. The resource file also uses a python based keywords from the library *CustomLibrary.py* for more specialised functionality. Countless community made libraries exist that extend Robot Framework in various ways!

```
*** Settings ***
Documentation     A test suite for valid login.
...
...               Keywords are imported from the resource file
Resource          keywords.resource
Default Tags      positive

*** Test Cases ***
Login User with Password
    Connect to Server
    Login User            ironman    1234567890
    Verify Valid Login    Tony Stark
    [Teardown]    Close Server Connection

Denied Login with Wrong Password
    [Tags]    negative
    Connect to Server
    Run Keyword And Expect Error    *Invalid Password    Login User    ironman
    Verify Unauthorised Access
    [Teardown]    Close Server Connection
```

Figure 4: Robot Framework requires the test cases in the form of scripts; some examples can be tried out directly on the project website.



What is RobotMK?

RobotMK contains the names of the two tools **Robot Framework** and **CheckMK** (for further details see below).

With **Robot Framework** you can automatically test programs, web sites (and much more...) from the perspective of the user.

Software developers appreciate this type of tests, as an application can thus for instance be

Figure 5: The RobotMK add-on works with CheckMK version 1.6 or newer.

real user monitoring is therefore combined with synthetic monitoring, which is preferably used during development or before the release of a new version. For example, if the shopping cart fails to respond within a second during synthetic monitoring, the customer does not even get to see the software. On the one hand, performance problems and errors can be detected at an early stage; on the other hand, performance guidelines and service-level agreements (SLAs) can be ensured in advance.

Once the application has been deployed, RUM then ensures that the application runs as expected during operation. You need RUM to reveal sudden failures or performance problems with third-party services, as well as discover how the application behaves on users' devices, especially if they are using some kind of exotic smartphone.

Reaching Your Goals

If you are now considering E2E monitoring, you should pause for a moment and plan how you are going to use it up front. It is the only way to ensure that the monitoring system will monitor the really important functions in the application in a targeted and effective way, and the only way to choose a suitable solution.

The first step is to determine which goals you want to achieve with monitoring. For example, is the shopping cart response time particularly important for the company's success? It makes sense to formulate specific business goals (e.g., getting more customers to complete a purchase or generating at least EUR10,000 in sales per hour).

The next step is to find out which metrics you need to achieve your goals and to what extent you need them. You need to consider not just the measuring points within the application; you also need to link the results of E2E monitoring with other data, such as the current server load. This approach is the only way to find out whether sluggish shopping cart response is due to an overloaded Apache web server or a poor implementation. Identifying the required data sources and merging them skillfully is important. In larger companies, several teams of developers, DevOps personnel, and administrators might even have to collaborate.

Choice of Tools

When choosing a specific monitoring solution, you need to make sure it meets your objectives and requirements. Checking whether existing test and monitoring tools are suitable for RUM or synthetic monitoring might mean you do not need to introduce new solutions. Ideally, the monitoring solution will take care of version management for all metrics, test cases, and configuration.

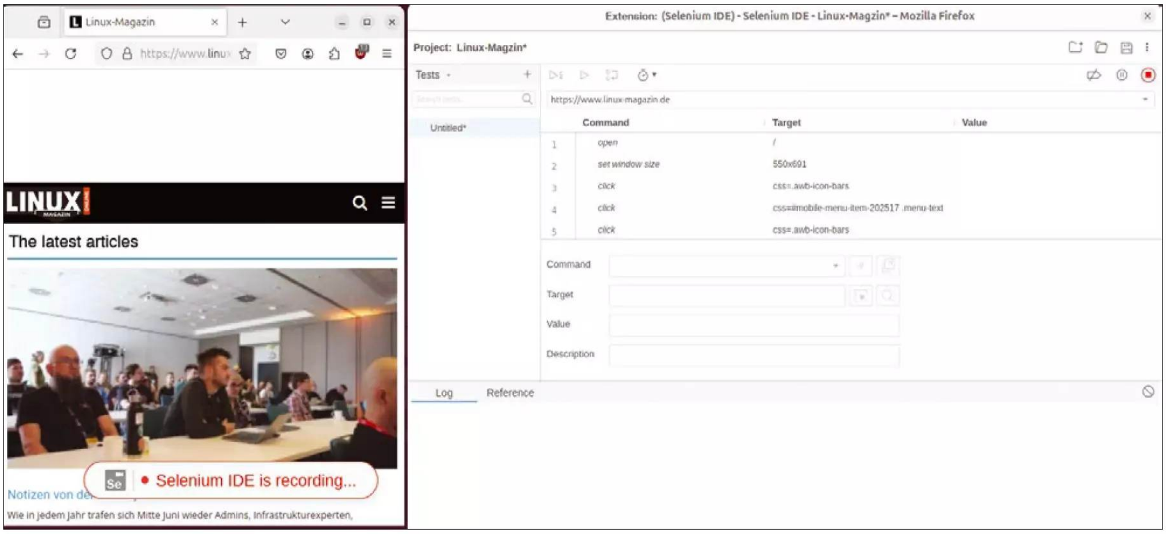
For RUM, you need to integrate your choice of monitoring tool into the application. An SDK designed for this purpose will save you some work.

Setting the measurement points to suit your requirements will also keep the noise to a minimum. Of course, the monitoring solution has to save the measurement data it generates somewhere. If possible, use an open format, which makes it easier to evaluate, store, and share the data later.

You also need to clarify which data storage and databases can be used. For example, if you want the software to track each individual customer through the check-out process in your online store, large volumes of data will accumulate in a short period of time; you need to collect and store the data in the background with the use of a suitable infrastructure.

Operational Management

When new functions are introduced or after completing work on the source code, you have a huge risk of performance problems and bugs. If at all possible, it makes sense to run each new version of the application through E2E monitoring internally


Linux-Magazin
https://www.linu
LINUX MAGAZIN
The latest articles
Selenium IDE is recording...
Notizen von de
Wie in jedem Jahr trafen sich Mitte Juni wieder Admins, Infrastrukturexperten,
Extension: (Selenium IDE) - Selenium IDE - Linux-Magzin* – Mozilla Firefox
Project: Linux-Magzin*
Tests
Untitled*
https://www.linux-magazin.de
Command
Target
Value
Description
Log
Reference

	Command	Target	Value
1	open	/	
2	set window size	550x691	
3	click	css=.awb-icon-bars	
4	click	css=#mobile-menu-item-202517 .menu-text	
5	click	css=.awb-icon-bars	

Figure 6: Selenium can replay recorded actions at different speeds.

before it is released, so you can rule out performance problems up front. All tests and RUM need to take place under real conditions and, above all, on various user devices. In other words, do not simply test the monitoring solution with a web application in the widely used Chrome browser, but at least in Firefox, as well. Moreover, the requests must originate from different locations. For example, you should test whether access to your online store from provider networks A and B is equally quick. Finally, you should always question whether the values output by your monitoring solution are focused correctly. For example, is the shopping cart performance still important after two years, or do you have different goals now?
Finally, you also want to monitor the E2E monitoring solution. A failed solution will not reveal problems or measure performance. E2E monitoring by no means makes classic monitoring systems superfluous. You still need to be notified without delay if the machine with the web server suddenly crashes, and an E2E monitoring tool cannot detected this kind of error.

Conclusions

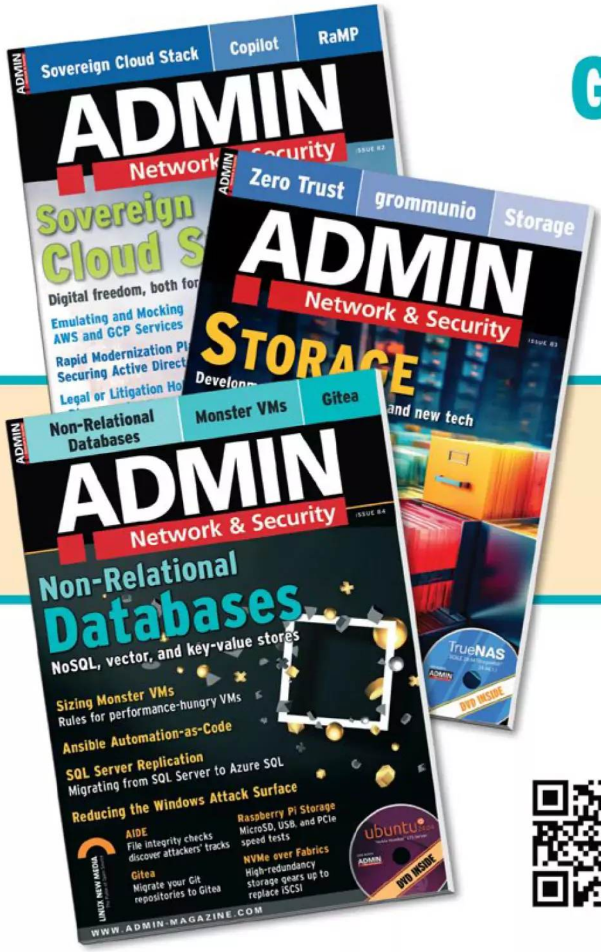
End-to-end monitoring focuses on monitoring the entire application. The data provided helps improve the user experience and performance in particular. Additionally, E2E monitoring detects malfunctioning features that conventional monitoring does not. E2E monitoring is particularly beneficial for applications that comprise several components or are based on a microservices architecture by ensuring that all components interact correctly in the background. This monitoring solution therefore covers areas that conventional monitoring does not and should always be in place, especially for web applications. ■

Info

[1] Grafana Faro: [https://grafana.com/oss/faro/]
[2] Basic RUM: [https://www.basicrum.com]
[3] OpenTelemetry-Go: [https://github.com/open-telemetry/opentelemetry-go]
[4] Robot Framework: [https://robotframework.org]
[5] RIDE: [https://github.com/robotframework/RIDE]
[6] RobotMK: [https://www.robotmk.org]
[7] CheckMK synthetic monitoring: [https://checkmk.com/product/synthetic-monitoring]
[8] Zabbix synthetic monitoring: [https://www.zabbix.com/documentation/1.8/en/manual/web_monitoring/real_life_scenario]
[9] Cypress: [https://www.cypress.io]
[10] Selenium: [https://www.selenium.dev]

The Author

Tim Schürmann is a freelance computer scientist and author. Besides books, Tim has published various articles in magazines and on websites.



GET TO KNOW ADMIN

ADMIN is packed with detailed discussions aimed at the professional reader on contemporary topics including security, cloud computing, DevOps, HPC, containers, networking, and more.

Subscribe to *ADMIN* and get 6 issues delivered every year

ADMIN Network & Security magazine is your source for technical solutions to real-world problems.

@adminmagazine @adminmag ADMIN magazine @adminmagazine

Four solutions for Prometheus long-term storage

Titans

If you use Prometheus as a time series database, you will know that the more data it stores, the slower it becomes. Thanos, Cortex, Mimir, and M3DB set out to solve this problem in totally different ways. We reveal the candidates' strengths and weaknesses. By Martin Loschwitz

One key difference between current IT setups and those of the past is that current setups are typically designed for scalability. Computing power, RAM, disk space – all can ideally be added or removed dynamically in a state-of-the-art setup. For ongoing operations, this results in a few challenges that administrators were unlikely to confront until the advent of OpenStack, Ceph, Kubernetes, and the like. They include, for example, the need to know when it's the right time to add additional resources. "When they are needed" is the easy answer, because that directly prompts the next questions: When exactly are new resources needed, and how do admins work around bottlenecks and delays in delivery of the new equipment?"

Trending software helps answer these questions early enough that you can plan in peace and sleep well at night. Under the hood, you will often find a time series database that collects and saves metrics data from all instances of the entire setup at regular intervals before correlating the acquired data and alerting if defined threshold values are exceeded. Right now, Prometheus is probably the most widely used of the candidates in this field.

If you want to scale or store data in the long term for analysis, four open source approaches stand out from the crowd:

1. Thanos is a cluster solution with built-in long-term storage.
2. Cortex makes it easy to add a genuine long-term storage engine to Prometheus.
3. Mimir, on the other hand, is a Cortex fork that is now very different from its ancestor and has its own functions.
4. M3DB is a converter between the worlds, combining central design elements of both Thanos and Cortex.

You therefore need to make a decision. It makes sense to answer three questions first:

- Which solution is best suited to your application?
- What are the individual strengths and weaknesses of the tools?
- Which is the easiest to manage on a day-to-day basis?

In this article, I take a detailed look at the four test candidates and reveal what you can look forward to or need to worry about.

Prometheus

Prometheus is a classic time series database supported by an Alert Manager that evaluates metrics data. Originally created by SoundCloud, Prometheus can be found in scalable environments worldwide. Additionally, Prometheus is usually teamed with Grafana, which reads the metrics from Prometheus and conjures up attractive and, above all, easy-to-understand graphs (Figure 1). Everything is great, you might think.

Battle-hardened Prometheus admins will probably disagree at this point because, as good as Prometheus might be for collecting basic data for trending and supporting monitoring on the basis of individual events, the work of keeping historical metrics data for a really long time to identify trends can soon become tedious. In the Prometheus context, a permanent problem is that it becomes slower and slower the more historical data it has to keep in its repository.

Although the situation now is far better than it was just a few years ago, anyone who challenges Prometheus to store historical metrics

Lead Image © sidewaypics, 123RF.com

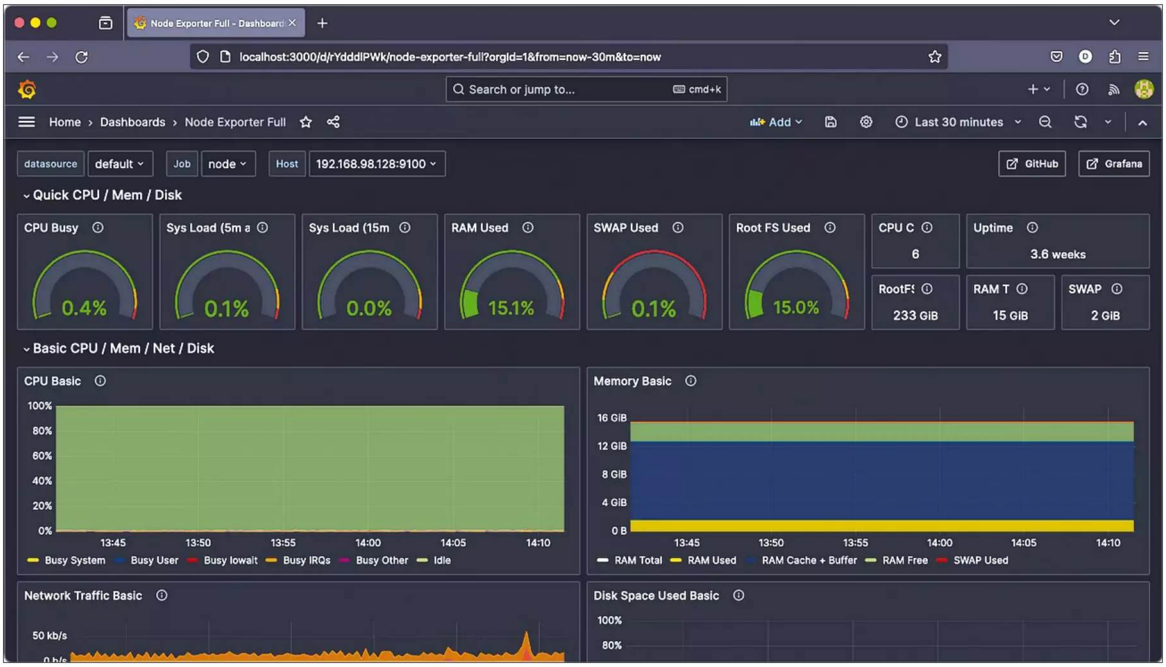

Node Exporter Full - Dashboard
localhost:3000/d/rYdddlPWk/node-exporter-full?orgId=1&from=now-30m&to=now
Search or jump to...
cmd+k
Home › Dashboards › Node Exporter Full
Add
Last 30 minutes
datasource
default
Job
node
Host
192.168.98.128:9100
GitHub
Grafana
Quick CPU / Mem / Disk
CPU Busy
0.4%
Sys Load (5m a
0.1%
Sys Load (15m
0.0%
RAM Used
15.1%
SWAP Used
0.1%
Root FS Used
15.0%
CPU C
6
Uptime
3.6 weeks
RootF:
233 GiB
RAM T
15 GiB
SWAP
2 GiB
Basic CPU / Mem / Net / Disk
CPU Basic
100%
80%
60%
40%
20%
0%
13:45
13:50
13:55
14:00
14:05
14:10
Busy System
Busy User
Busy Iowait
Busy IRQs
Busy Other
Idle
Memory Basic
16 GiB
12 GiB
8 GiB
4 GiB
0 B
RAM Total
RAM Used
RAM Cache + Buffer
RAM Free
SWAP Used
Network Traffic Basic
50 kb/s
0 b/s
Disk Space Used Basic
100%
80%

Figure 1: **Grafana is considered to be the jack of all trades when it comes to data visualization and is often used in conjunction with Prometheus; however, if you want to scale Prometheus or store data there in the long term, you need a helper for the time series database.** © Grafana Labs

from a large-scale setup over several years is unlikely to enjoy working with the solution. Problematic architectural decisions that date back to the early days of the application and are difficult or impossible to correct retrospectively play a major role here.

Thanos

To get the show started, I looked at Thanos (Figure 2), which is one of the oldest candidates to create a cluster layer for Prometheus [1]. It not only takes care of storing data over a long period of time, but also effectively seeks to make Prometheus capable of high availability. By default, the time series database is a little bare bones in this respect. The Prometheus developers just assume that users can simply roll out several Prometheus instances at the same time if redundancy is required.

Because Prometheus itself pulls metrics data from defined targets, the active Prometheus instances could quite simply permanently scan everything

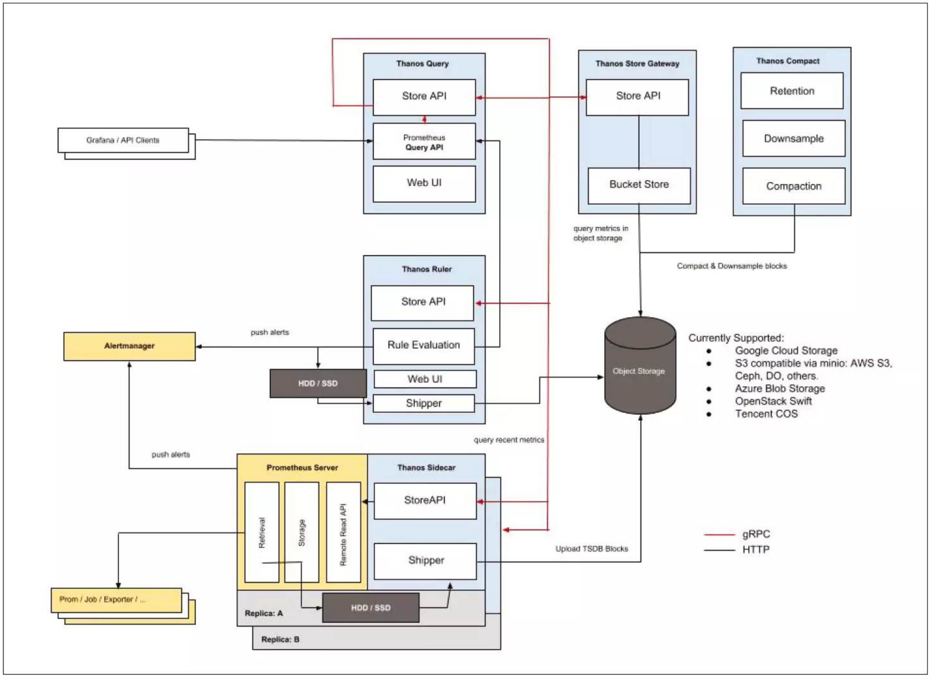

Thanos Query
Store API
Prometheus Query API
Web UI
Grafana / API Clients
Thanos Store Gateway
Store API
Bucket Store
Thanos Compact
Retention
Downsample
Compaction
query metrics in object storage
Compact & Downsample blocks
Thanos Ruler
Store API
Rule Evaluation
Web UI
Shipper
push alerts
Alertmanager
HDD / SSD
Object Storage
Currently Supported:
Google Cloud Storage
S3 compatible via minio: AWS S3, Ceph, DO, others.
Azure Blob Storage
OpenStack Swift
Tencent COS
query recent metrics
push alerts
Prometheus Server
Retrieval
Storage
Remote Read API
Thanos Sidecar
StoreAPI
Shipper
Upload TSDB Blocks
gRPC
HTTP
Prom / Job / Exporter / ...
Replica: A
HDD / SSD
Replica: B

Figure 2: **Thanos is one of the oldest scalability solutions for Prometheus and promises a global view of all data and fast long-term data storage.** © Thanos

in parallel. However, this arrangement would increase the load on the data sources and mean no global, standardized query view for metrics data. Thanos retrofits this view, collecting metrics data from different Prometheus instances (or Prometheus instance pairs), deduplicating the data, and finally storing the results permanently.

Thanos relies on a number of components. The application first provides each Prometheus instance with a sidecar (i.e., a service that connects to Prometheus and extracts incoming data from it). The sidecar sends the data to an object store, typically a Simple Storage Service (S3). Of course, this does not have to be the real Amazon S3; a local replica based on, say, MinIO or RADOS Gateway will work just as well. The database supports a Compactor component that regularly reads the stored metrics data, downsamples the data according to defined rules, and then stores everything in the object store. A query component is available for external queries, which is where the global view is implemented. When you query metric data with `thanos query`, you are querying the entire database.

All Thanos components scale horizontally, and you can deploy an arbitrary number of query instances, although you will then need to set up a load balancer upstream. Last but not least, Thanos comes with a component known as the Ruler, which primarily communicates with the Prometheus Alert Manager, or more likely with the alert managers of several Prometheus instances or Prometheus high-availability pairs. In this way, Thanos implements a global alert view across all Prometheus instances, including deduplication.

Database queries by `thanos query` use the same syntax that Prometheus itself uses. In other words, the query language is PromQL. From the administrator's point of view, a useful option opens up of defining Thanos instead of Prometheus as the data source in Grafana, which simplifies handling in Grafana: Instead of scanning several Prometheus instances for data and then combining them in a dashboard, Grafana simply retrieves the metrics data from Thanos, which has already created a global view.

Additionally, according to the community's experience, Thanos works extremely quickly in small, medium, and large setups and delivers high performance.

Difficult to Install

The downside of Thanos is that deploying the tool turns out to be far more complicated than it should be. For example, if you want to roll out Thanos along with Prometheus in Kubernetes, you will find an option in the Prometheus Kubernetes operator that lets you roll out Prometheus along with the Thanos sidecar, but you will have to set up the Thanos services more or less manually, because you will not find ready-made Helm charts by the authors or instructions to help you do this.

The solution is not up to date in this respect; in fact, it could be far better. However, once you have Thanos assembled to suit your requirements, you can look forward to a powerful and very reliable method of clustering Prometheus and to a global data view across multiple instances of the time series database.

M3DB

M3 just goes to show how widespread enterprise IT has become (Figure 3). M3 was developed by Uber, a provider of taxi and rental car services; its aim from the outset was to work around the lack of scalability in Prometheus and support efficient long-term data storage [2]. The associated terminology might cause some confusion – the entire solution operates online as both M3 and M3DB. M3DB is just the underlying distributed time series database for (i.e., an element of) M3.

M3DB comprises storage and seed nodes that are basically storage nodes but also perform cluster management for M3DB and enforce a quorum. From the perspective of the other M3 components, though, M3DB looks like a single component, especially because data does not flow directly into it. Instead, the M3 coordinator handles this. From the admin's point of view, the most important thing is that M3DB is a completely separate database with its own data storage, which is a key difference from the Thanos design, which does not have its own time series database but instead stores the data as a blob in object memory.

In principle, M3 Coordinator is a kind of sidecar that sits next to every Prometheus instance acquiring data that it then sends to M3DB in parallel, making it very similar to the Thanos sidecar. Like the sidecar in Thanos, you need to roll out M3 Coordinator so that it uses the Prometheus Read API to read metrics data directly from Prometheus and forward it. Of course, M3 Coordinator also scales horizontally, which is how its authors avoid bottlenecks.

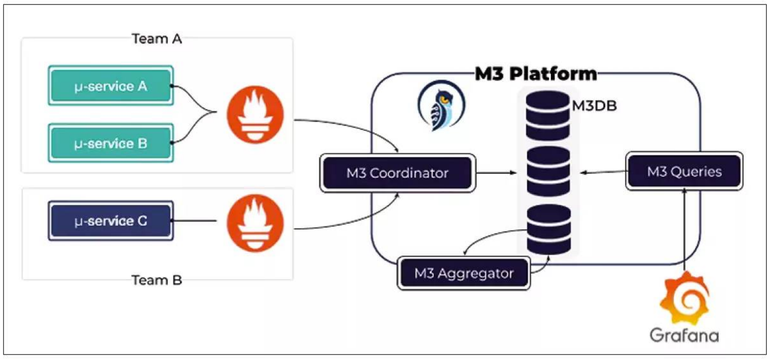


Figure 3: M3 comes from Uber and is similar in its architecture to both Cortex and Thanos. The solution is neatly integrated with Kubernetes, but it is not very well known, all told. © Uber

That said, deploying the system in Kubernetes is nearly as complex as Thanos. Although Helm charts exist for M3, the official Prometheus operator for Kubernetes does not have an option for rolling out the M3 sidecar in the form of the Coordinator along with Prometheus. In return, the deployment of M3 in Kubernetes is much easier because of the existing operator and Helm chart. Again, the situation with M3 is about as far from ideal as it is with Thanos.

M3 comes with two further components: the Aggregator and a query component. The Aggregator regularly scours the database according to rules defined by the admin and discards any data that is no longer needed. It also deduplicates data records (e.g., if they originate from high-availability setups with several Prometheus nodes that all monitor the same targets).

At the end of the day, the query service does exactly what it says on the box: It offers the option of centrally querying the data stored in M3DB by PromQL syntax and implements the kind of global view that Prometheus does not have by doing so.

Like Thanos, M3 is ideal for operating Prometheus in a horizontally scaling setup without having to create a number of different endpoints for using and managing the monitoring system. As with Thanos, you will find a couple of weaknesses in terms of M3 deployment, especially in the Kubernetes context. Unfortunately, M3 more or less fails to turn up when it comes to alert management. The tool largely ignores the Prometheus Alert Manager, forcing anyone who wants to deduplicate alerts in high-availability setups from Prometheus and forward them in the right way to use the Prometheus Alert Manager's onboard tools. Although this technique works, it is anything but convenient and feature rich.

Cortex

Unsurprisingly, Uber was not the only one to realize that Prometheus was an attractive solution but did not scale well. Shortly after Thanos, Cortex saw

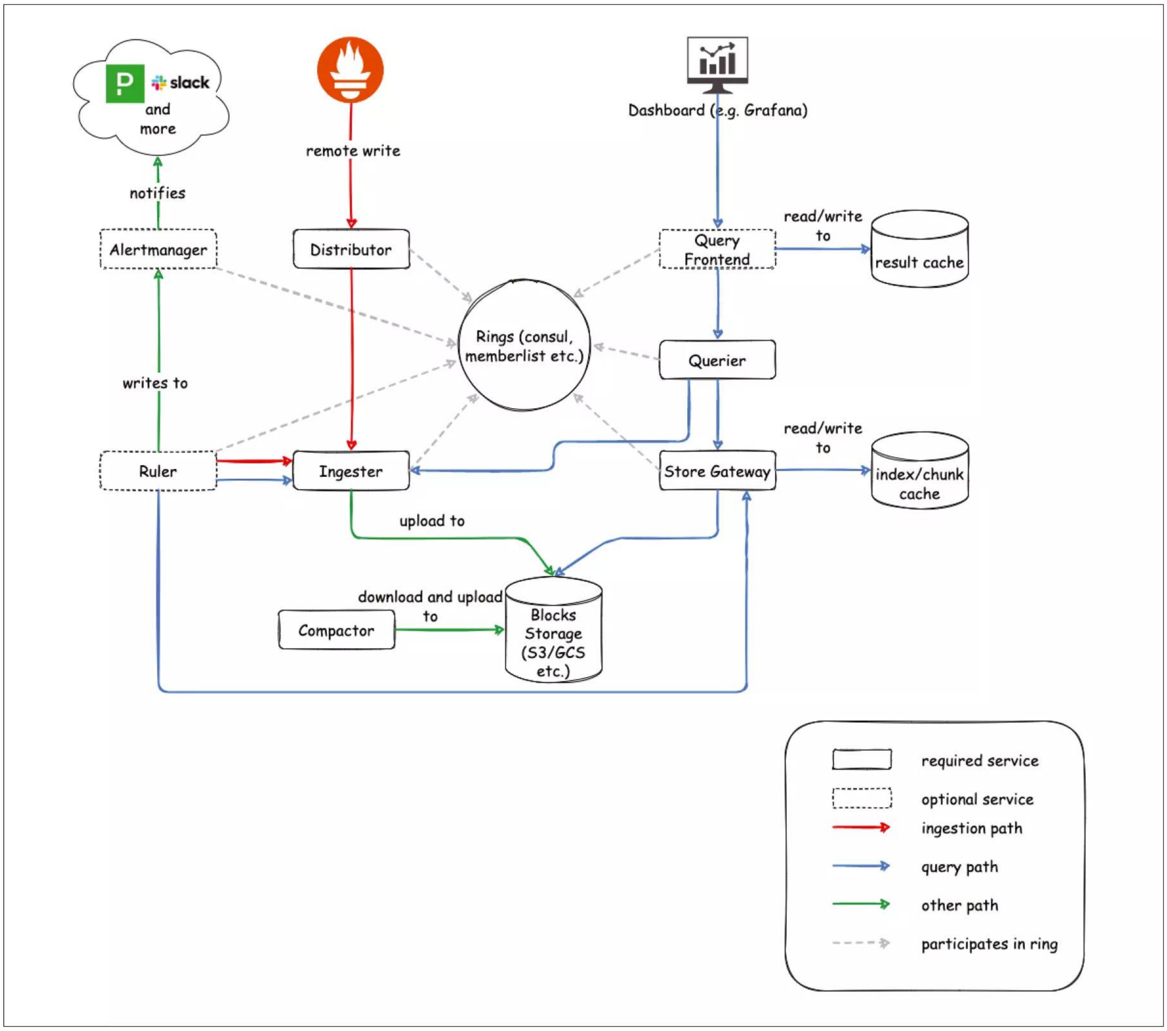

slack
and
more
notifies
remote write
Dashboard (e.g. Grafana)
read/write
to
result cache
Alertmanager
Distributor
Query
Frontend
Rings (consul,
memberlist etc.)
Querier
writes to
read/write
to
index/chunk
cache
Ruler
Ingester
Store Gateway
upload to
download and upload
to
Compactor
Blocks
Storage
(S3/GCS
etc.)
required service
optional service
ingestion path
query path
other path
participates in ring

Figure 4: **Cortex is like a hybrid of Thanos and M3, at least in terms of architecture. Cortex does not need a sidecar for Prometheus. Instead, it acts as a write engine in the background.** © Cortex

the light of day (**Figure 4**); although it aimed to solve the same problem, the approach came from a different direction. The strengths of Cortex [3] are the cloud-native-first principle and the resulting virtually seamless integration with Kubernetes. But first things first.

Under the hood, the Cortex architecture has clear similarities with M3. Cortex itself is also a time series database and contains all the components required for a distributed database. Interestingly, though, the data is stored in the background. In this respect, Cortex is more like Thanos. Cortex has no database service as such; instead, the data is stored in the object store on the back end, which, in turn, offers benefits for setups such as MinIO or the RADOS Gateway that already have S3-compatible storage in place. The data passes through several levels in Cortex before reaching the store.

Another remarkable thing about Cortex is how the tool accesses its data. Unlike Thanos or M3, Cortex does not connect to Prometheus as a client. Instead, the administrator configures a kind of internal Cortex load balancer, known as the Distributor, as a remote storage target in Prometheus. For some time now, Prometheus has offered the option of forwarding the data it collects to other locations, and Cortex makes extensive use of this ability. The Distributor works in conjunction with the Cortex Ruler, which determines which of the incoming data has to be stored, how, and for how long. The data is then passed through a component known as the Ingester to the configured object store in the background.

The Ingester is also wired to the Ruler component and filters out some of the incoming data at this early stage. The Cortex Compactor performs a similar task to that of the M3 Aggregator, cleaning up the stored data according to rules in the Ruler component, deduplicating the data, and finally writing the cleaned-up results to the store.

Cortex sets great store on options for querying the stored data. No fewer than four components make up the solution's query stack: An API relies on the PromQL interface to expose the service to the outside world. In Cortex speak, this is known as the Query Frontend. The front end keeps its own cache, which significantly accelerates queries for the same data occurring in the short term.

Front ends send queries to the Query component, which communicates with the Ruler to obtain additional information on the stored data but otherwise forwards the request to the store gateway. The gateway looks for the required data in the object store on the back end and finally delivers the results.

The store gateway also has its own cache to facilitate queries. Every single component in Cortex is implicitly redundant and scalable. Several instances of the Query Frontend can be used, which also serves as the endpoint for communication with visualizers such as Grafana, as well as the Ruler, querier services, or the store gateways. Of course, Cortex also offers a global view; that is, it can query metrics data across the boundaries of individual Prometheus instances or local high-availability setups from multiple Prometheus nodes.

Unlike the M3 developers, the Cortex developers did not ignore the Prometheus Alert Manager, which can be tied in directly to the Ruler component that then deduplicates incoming alerts and relays them to the outside world in various ways. Native integration components for services such as Slack, PagerDuty, and many others are also available.

Cortex's integration with Kubernetes is equally well solved. Although it has no connection in Prometheus Operator, for example, this lack is not too critical, because you can connect Cortex as a storage target in the Prometheus configuration. Prometheus Operator provides the needed switches and levers. Seamless Kubernetes integration for Cortex itself exists in the form of ready-to-go Helm charts, which means the entire service can be rolled out with relatively little overhead. If you want scalable, fast, long-term storage for Prometheus quickly, right from the outset on Kubernetes, Cortex is your best bet.

Mimir

The last candidate in the test is Mimir (**Figure 5**). The technical review of this time series database [4] does not need to be too detailed – Mimir and Cortex are largely similar, and that is no coincidence. Historically, the two products originate from Grafana Labs, which started to work on Cortex first and continued for several years but then forked off the Mimir project. Things don't always run smoothly in the open source scene, and the fork that turned Cortex into Cortex and Mimir is a good example. The decisive factor at the time was primarily the fact that Cortex development was now largely dominated by AWS developers and their own ideas of what a scalable Prometheus could or should look like.

As is so often the case in the community, at some point, people argued about the tiniest details right down to the level of individual commits. Ultimately, the former Cortex community split into two parts, which from the admin's point of view is quite significant: To this day, the development of Cortex is primarily driven by AWS, whereas Mimir's development is primarily driven by Grafana Labs.

The vast majority of admins would intuitively see this as an advantage: With a company like AWS, it cannot be assumed that it will go bust or simply disappear from the market through a buyout. However, appearances can be deceptive. Large corporations such as Google, Microsoft, and AWS have proven countless times in the past that they have no qualms about killing off individual tools or entire departments with the stroke of a pen if the figures don't add up. If AWS decided to ditch the team working on Cortex, the tool would lose the majority of its active developers overnight.

Mimir is a different story: Compared with the giant oil tanker that is AWS, Grafana is a rowboat without a hull full of tools to manage and develop. Mimir is of strategic importance for Grafana, which also offers Mimir as a hosted service. At the same time, Grafana is now too big to run into serious financial difficulties in the foreseeable future. It's unlikely that Mimir would lose the driving force behind its development for the time being, and some admins may be more inclined toward Mimir than Cortex.

Technically, the differences are marginal. Mimir works architecturally like Cortex; it integrates perfectly with Kubernetes thanks to Helm charts and can be set up quickly and easily, which is largely due to the streamlining of Mimir by Grafana. Large parts of Cortex's legacy codebase are no longer included in Mimir, simply because they are obsolete in current setups. Unlike Cortex, Mimir comes as a single binary that contains and can offer all the required functions.

The Mimir developers also claim better performance and easier handling than with Cortex for special data types such as cardinality handling. However, this niche application will not persuade most admins to vote for Mimir. More significant is the native scheduling component added to the query component that selects the appropriate instance for querying when required, although it is only noticeable in very large setups. Anyone who has decided on the basic functional principle of Cortex or Mimir and has ruled out Thanos or M3 will want to evaluate the two solutions extensively. Ideally, the main focus should be on the peculiarities of the solutions in the context of the respective deployment scenario.

Conclusions

A scalable, distributed Prometheus that works well and reliably is possible. All four test subjects in this comparison deliver impressive results, even if they rely on different architectures. As a veteran of the scene, Thanos scores points with its versatility but forces you to make some concessions when it comes to integration with Prometheus and Kubernetes. M3 offers good Kubernetes integration and is fast, but you will have to set it up manually as a separate service in a Kubernetes cluster along with Prometheus.

Cortex and Mimir are closely related, and the decision for or against one of the two solutions is more of a philosophical and strategic decision than technical one. Both tools integrate excellently with Kubernetes and, like Thanos and M3, offer a truly global view of all stored metrics data, including fast, long-term data storage. However, Mimir comes with the more modern and significantly paired-down codebase and has more momentum within the community – at least for the time being.

No matter which of the four approaches you choose after appropriate tests, none of the four representatives is a bad choice. Once again, the open source community very impressively demonstrates that it is ideally positioned for creating enterprise software and enterprise setups. ■

Info

[1] Thanos: [https://thanos.io]
[2] M3: [https://www.uber.com/en-DE/blog/m3]
[3] Cortex: [https://www.cortex.io]
[4] Mimir: [https://grafana.com/oss/mimir]

The Author

Martin Loschwitz is the founder and managing director of True West IT Services GmbH, which offers scalable IT infrastructure based on OpenStack and Kubernetes.

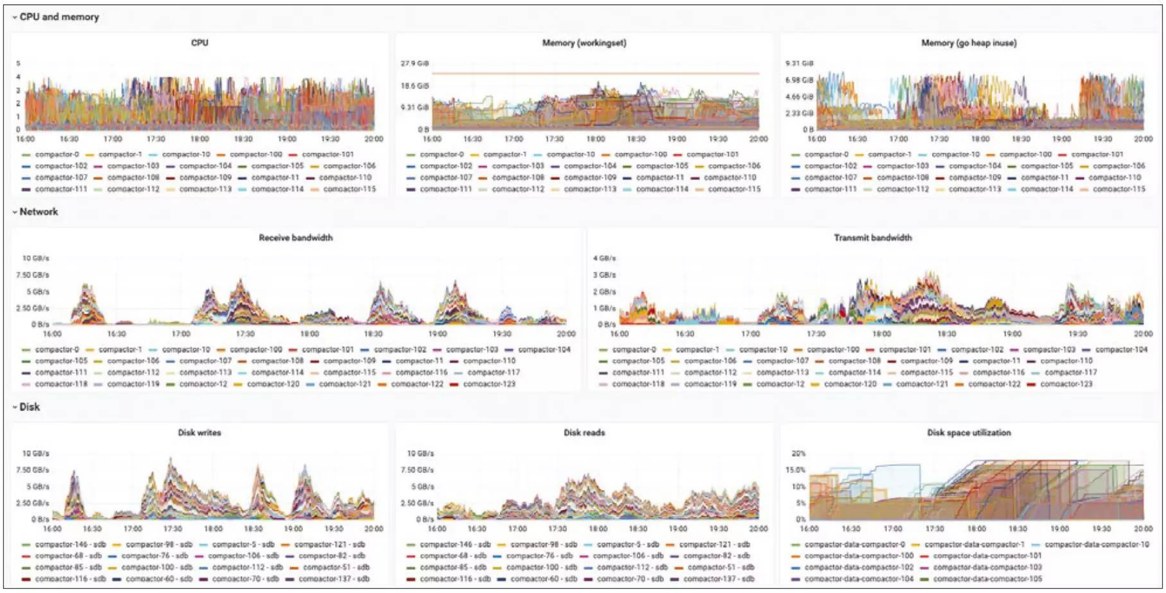
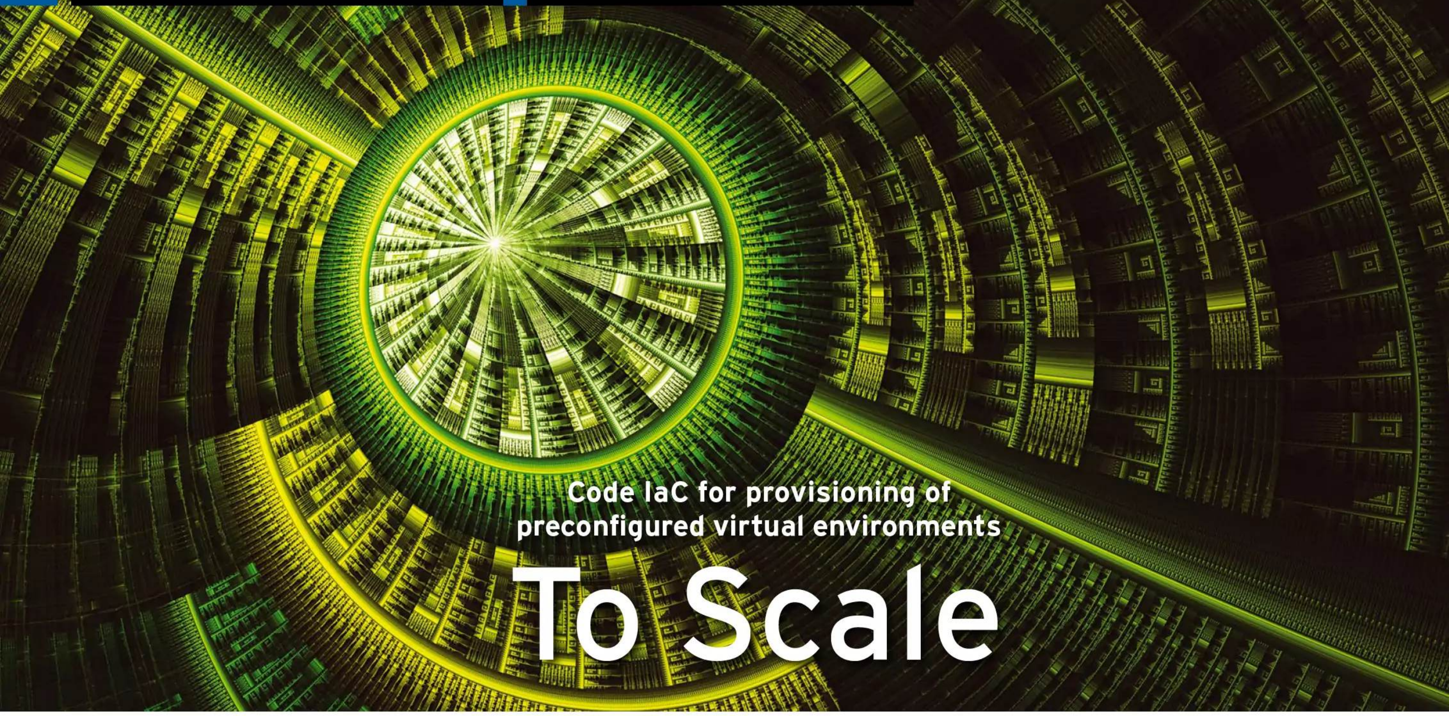


Figure 5: Grafana Labs says that the hosted version of Mimir is now handling more than a billion active data series and that Mimir can cope with this volume without any problems. © Grafana Labs



Code IaC for provisioning of preconfigured virtual environments

To Scale

We explore automation techniques for the Azure Dev Box cloud service.

By Kevin Wittmer

Microsoft Dev Box is a powerful Azure cloud-based service that offers self-service access to create preconfigured virtual environments. Developers, engineers, and technology-savvy professionals can quickly spin up virtual environments tailored to IT, developer, or business project needs with the use of project-specific templates and shared configurations. This service can eliminate the traditional challenges of provisioning and maintaining development or engineering environments, enabling teams to focus on innovation and productivity.

Azure Dev Box leverages Azure's virtual machine (VM) and imaging infrastructure to provide a secure, scalable, and flexible cloud platform service designed to accelerate the onboarding and provisioning of user-dedicated virtual desktop environments for various technical users.

Developers can access preconfigured, project-specific dev boxes, and the user-specific dev box environment can be further customized through YAML-based tasks, which can be shared in a GitHub catalog and applied during the final stage of creation or uploaded manually.

IT administrators have the option of using Microsoft Intune to manage dev box environments at scale, which ensures consistency, compliance, and streamlined operations. With seamless integration into Azure services and hybrid cloud connectivity to enterprise networks, Microsoft Dev Box is a good choice for organizations looking to improve collaboration and reduce onboarding time for IT projects and users alike.

Dev Box Architecture

The top-level abstraction in the Microsoft Azure Dev Box platform architecture and a critical component for setting up projects, provisioning environments, and managing their lifecycle is Microsoft Dev Center (Figure 1). For developers, engineers, and business power users, each project within Dev Center serves as the starting point for creating customized virtual environments. These projects are linked to a host pool (also known as a dev box pool), which provides the Azure runtime host context for VM instances [1].

The dev box pool uses a predefined dev box definition for the initial sourcing of compute and storage settings and allows you to specify network connectivity, which can be a customer-provided virtual network and subnet or a Microsoft internal network. When setting up your initial dev box, confirm that you have the necessary Windows licenses, which may be associated with a Microsoft Windows Enterprise licensing structure. Refer to Microsoft's quick start documentation for more detailed information about licensing considerations affecting a dev box deployment [2].

Dev box projects can be aligned with different environments, such as development, QA, or production, each linked to a specific Azure subscription. IT organizations often map these environments to separate subscriptions and apply DevOps activities across subscription groups directly associated with a given IT product, service, or solution. Microsoft's VM image infrastructure supports the Dev Box self-service virtualization model, including the Compute Gallery and image template facilities, which enable the creation and deployment of custom images. Azure VM Image Builder uses these image templates to create VM images for Dev Box, although the process takes some time to complete as a cloud workload.

Automation Toolbox

Although Microsoft offers a web-based user interface (UI) for creating Azure resources (Figure 1), in this article I focus on introducing and demonstrating infrastructure-as-code (IaC) techniques for creating and managing Azure resources like a dev center. To achieve this, you will use template

Lead Image © vladnikon, 123RF.com

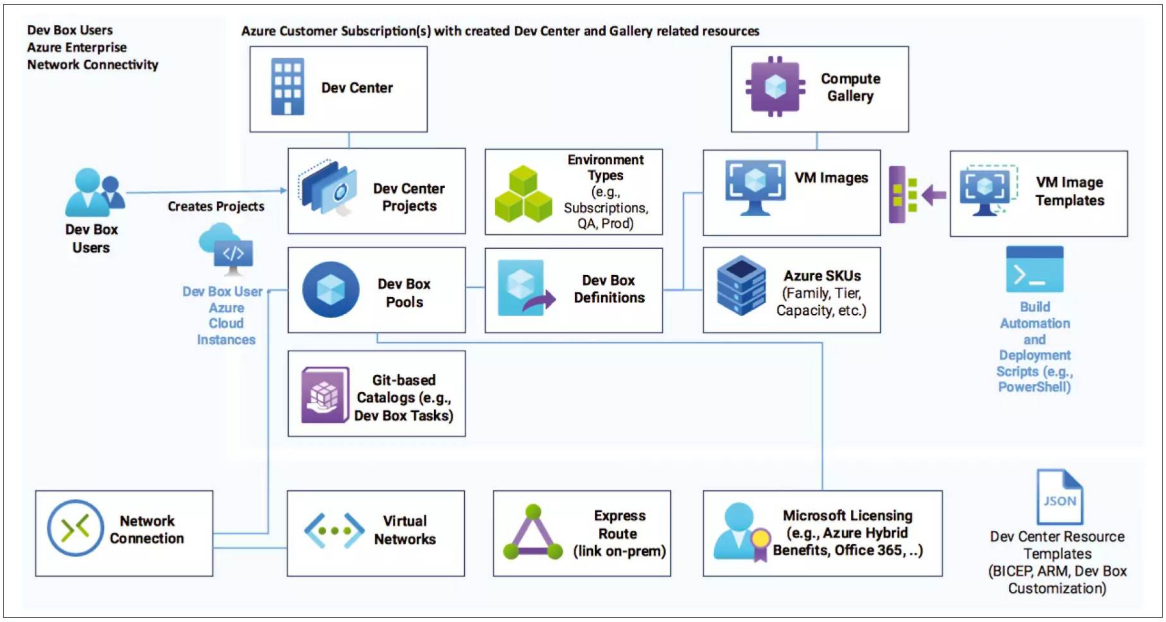

Dev Box Users
Azure Enterprise
Network Connectivity
Azure Customer Subscription(s) with created Dev Center and Gallery related resources
Dev Center
Compute Gallery
Dev Box Users
Creates Projects
Dev Center Projects
Environment Types (e.g., Subscriptions, QA, Prod)
VM Images
VM Image Templates
Dev Box User Azure Cloud Instances
Dev Box Pools
Dev Box Definitions
Azure SKUs (Family, Tier, Capacity, etc.)
Build Automation and Deployment Scripts (e.g., PowerShell)
Git-based Catalogs (e.g., Dev Box Tasks)
JSON
Network Connection
Virtual Networks
Express Route (link on-prem)
Microsoft Licensing (e.g., Azure Hybrid Benefits, Office 365, ..)
Dev Center Resource Templates (BICEP, ARM, Dev Box Customization)

Figure 1: Dev Box architecture resource components.

Table 1: BICEP Resource Definitions

Resources Versioned w/@yyyy-mm-dd-state	Common and Resource-Specific Properties	Description
`Microsoft.DevCenter/devcenters`	Identity (e.g., `userAssignedIdentities`), `location`, `name` `devBoxProvisioningSettings`, `networkSettings`, `projectCatalogSettings`, etc.	The top-level resource that serves as a container for multiple projects. These projects are directly instantiable by the end user and can share common resources, like dev box definitions and network connections.
`Microsoft.DevCenter/devcenters/devboxdefinitions`	`hibernateSupport`, `imageReference`, and `osStorageType`; also, `sku` with `family`, `tier`, and `capacity` as key attributes	The definition of a dev box based on image type, `sku` attributes, and compute resource allocations.
`Microsoft.DevCenter/devcenters/galleries`	`parent`, `name`, and `galleryResourceId`	Associated gallery holding VM images.
`Microsoft.DevCenter/devcenters/attachednetworks`	`networkConnectionId`	The attached network connection to allow dev boxes to connect to existing virtual networks.
`Microsoft.DevCenter/devcenters/catalogs`	Git-related properties for GitHub and Azure DevOps with attributes that specify `branch`, `path`, `uri`, and `secretIdentifier`	The attached GitHub repos hold shared ancillary artifacts, such as dev box tasks across DevBox instances.
`Microsoft.DevCenter/devcenters/environmentTypes`	`displayName`	Environment types that map different Azure subscriptions (e.g., QA).
`Microsoft.DevCenter/networkConnections`	`subnetId`, `networkingResourceGroupName`, `organizationUnit`; also Active Directory domain-related properties, including: `domainJoinType`, `domainName`, `domainPassword`, and `domainUsername`	The Active Directory (AD) join type, name, and credentials; also, the `subnetId` to which you attach dev box VMs.
`Microsoft.DevCenter/projects/pools`	`devBoxDefinitionName`, `networkConnectionName`, and `licenseType` are key properties	A pool binds together the dev box definition, network connection, and license type; it also contains properties to further customize the compute, as well as aspects such as stop on disconnect.
`Microsoft.DevCenter/projects`	Includes `devCenterId`, `catalogSettings`, `description`, and `displayName`; this resource binds together many other resources in a parent/child relationship and serves as the leading top-level instantiable resource in the Dev Box user model: `Microsoft.DevCenter/projects/allowedEnvironmentTypes`, `Microsoft.DevCenter/projects/attachednetworks`, `Microsoft.DevCenter/projects/catalogs`, `Microsoft.DevCenter/projects/devboxdefinitions`, `Microsoft.DevCenter/projects/pools`, `Microsoft.DevCenter/projects/images`, `Microsoft.DevCenter/projects/environmentTypes`	A project can be defined to reuse key resource artifacts of Dev Box, including environment types, attached networks, Git-based catalogs, dev box definitions, dev pools, and associated images. The developer, user, or admin (e.g., think PowerShell or API) of Dev Box will create a new dev box from the properties of the dev center project.

languages and tools like Bicep, ARM, and PowerShell.

Microsoft provides a comprehensive set of Bicep-friendly resource definitions specifically designed for Dev Center resources, summarized in **Table 1**. This table outlines common and resource-specific properties required to codify resource creation and shape a custom Bicep file for dev box scenarios. Among these resources, dev box definitions and dev center projects require intensive variable mapping to link specific resource properties. Fortunately, there's no need to start from scratch – the GitHub *azure-quickstart-templates* project maintained by Microsoft is an excellent Bicep-based reference and starting point that helps you get up and running quickly **[3]**.

Deploying a `Microsoft.DevCenter` (DevCenter hereafter) instance requires creating a service principal or managed identity for script-based setup and IaC automation. You should also plan and assign users and groups with Dev Box User and Dev Box Admin roles to manage and access DevCenter resources. Other roles, such as the Contributor Gallery role, must be assigned to add VM images to the Azure Compute Gallery.

Codifying DevCenter

Coding IaC for a DevCenter service simplifies resource deployment and management while ensuring consistency and repeatability. Domain-specific IaC languages like Bicep make this process easier by abstracting the complexity of Azure Resource Manager (ARM) templates. With Bicep's clean, modular syntax, you can define key resources like `DevCenter`, `DevCenterGallery`, `networkConnections`, and `devboxdefinitions`. These resources make up the foundation of a dev box environment, linking resources such as VM images, virtual networks, and related configurations.

To put these resource definitions into action, start with a basic Bicep file skeleton to set up DevCenter and its related resources. During this initial phase, focus on key aspects, such as the name, parent, and `dependsOn` relationships. **Listing 1** offers this template structure as a starting point, with the next step being to codify an IaC DevCenter file iteratively by adding parameter variables and mapping them to the properties of the various Bicep resources for your specific context and use case.

When writing IaC code for DevCenter, it's important to remember that Microsoft regularly updates the API specification **[4]**, with monthly preview versions and full releases typically occurring annually (e.g., 2023-04-01, 2024-02-01). To stay compatible with the latest Azure features, Bicep uses versioned specifications marked by the @ symbol (e.g., @2024-02-01) that align resource definitions with specific ARM API versions. This method ensures smooth integration of new DevCenter capabilities as they are released.

The Bicep Visual Studio (VS) Code extension from the VS Code Marketplace simplifies creating Bicep resource definitions and property mappings. This extension includes IntelliSense for auto-completion and suggestions for a quick exploration of available resource definitions. The extension also supports dot property access, making navigating and mapping properties within resource definitions

Listing 1: Skeleton DevCenter File

```
resource devcenter 'Microsoft.DevCenter/devcenters@2024-02-01' = {
  name: devcenterName .. ..  }

resource devcenterGallery 'Microsoft.DevCenter/devcenters/galleries@2024-02-01' = {
  name: galleryName
  parent: devcenter
  ...
  dependsOn: [
    readGalleryRole
    contributorGalleryRole
  ]
}

resource networkConnection 'Microsoft.DevCenter/networkConnections@2024-02-01' = {
  name: networkConnectionName .. .. }

resource attachedNetworks 'Microsoft.DevCenter/devcenters/attachednetworks@2024-02-01' = {
  name: networkConnection.name
  parent: devcenter
  ... }

resource devcenterGalleryImage 'Microsoft.DevCenter/devcenters/galleries/images@2024-02-01' existing = {
  name: imageDefinitionName
  parent: devcenterGallery
  ... }

resource customDevboxDefinitions 'Microsoft.DevCenter/devcenters/devboxdefinitions@2024-02-01' = {
  name: customizedImageDefinition.name
  parent: devcenter
  ...
  dependsOn: [
    attachedNetworks
    imageTemplateBuild
  ]
}

resource project 'Microsoft.DevCenter/projects@2024-02-01' = {
  name: projectName
  ... }

resource customImagePools 'Microsoft.DevCenter/projects/pools@2024-02-01' = [for pool in devceterSettings.customizedImagePools: {
  name: pool.name
  parent: project
  ...
  dependsOn: [
    customizeDevboxDefinitions
    imageTemplateBuild
  ]
}]
```

easier. Additionally, it allows you to add and customize code snippets, which is helpful for repetitive tasks like defining DevCenter resources.

Listing 2 presents a Bicep template skeleton for creating custom Azure VM images. It starts by defining an Azure Compute Gallery, along with an image definition resource that serves as a grouping construct for images. The key resources for building these images are the image template – used by the Azure Image Builder service as its primary input – and the image build specification. Additional ancillary resources are required to manage image versions and the associated deployment script (check Microsoft docs for additional details).

The resulting custom image generated by the Image Builder service can later be linked to the dev box project with the dev box definition, allowing it to be used as a source for provisioning new virtual environments.

With an understanding of the dev box and image resources at hand, you can now explore key resources requiring substantial IaC to map properties to script-wide variables. The dev box definition is a central resource directly linking custom images with the `id` property in the `imageReference` field. Important properties include the `sku`, which controls resource specifications such as capacity, VM series (e.g., `Standard_D4s_v3`), `size`, and `tier` (e.g., `standard` or `premium`). Other key properties are `hibernateSupport`, which enables faster startup times through hibernation, and `osStorageType`, which defines the operating system disk type (e.g., `Standard SSD` or `Premium SSD`) to optimize performance. Listing 3 presents a complete dev box example that can support various use cases.

Custom Image Definition

Next, I dive deeper into a template for defining a custom image for a VM (Listing 4). This template customizes a source platform image by specifying the VM profile, including compute size, disk size, and user identity, while also injecting PowerShell instructions into the VM definition. Key elements include defining the VM profile attributes for compute (e.g., `Standard_DS2_v2`) and disk size (e.g., a 47GB OS disk). The source image details, such as `offer`, `publisher`, `sku`, and `version`, are also specified. Additionally, the template includes steps to install software by PowerShell and prepare the image for distribution in a gallery for sharing.

The distribution specification shown is important in managing how the image is shared. It defines the target gallery ID and specifies the Azure regions where the image will be replicated. A brief explanation of these

Listing 2: Gallery and Image Resource Artifacts

```
resource computeGallery 'Microsoft.Compute/galleries@2024-02-01' = {
  name: galleryName ...
}

resource imageDefinition 'Microsoft.Compute/galleries/images@2024-02-01' = {
  parent: computeGallery
  name: imageDefinitionName ...
}

resource imageTemplate 'Microsoft.VirtualMachineImages/imageTemplates@2024-02-01' = {
  name: imageTemplateName  ...
  dependsOn: [
    imageDefinition
  ]
}

resource imageTemplateBuild 'Microsoft.Resources/deploymentScripts@2024-02-01' = {
  name: imageBuildName ...
  }
  dependsOn: [
    imageTemplate
  ]
}
```

Listing 3: Dev Box Definition

```
resource symbolicname 'Microsoft.DevCenter/devcenters/
  devboxdefinitions@2024-02-01' = {
  parent: resourceSymbolicName
  location: 'string'
  name: 'string'
  properties: {
    hibernateSupport: 'string'
    imageReference: {
      id: 'string'
    }
    osStorageType: 'string'
    sku: {
      capacity: int
      family: 'string'
      name: 'string'
      size: 'string'
      tier: 'string'
    }
  }
  tags: {
    {customized property}: 'string'
  }
}
```

Listing 4: Image Template Definition

```
resource imageTemplate 'Microsoft.
  VirtualMachineImages/imageTemplates@2024-02-01' = {
  name: imageTemplateName
  location: location
  identity: { .. }
  properties: {
    buildTimeoutInMinutes: 180
    vmProfile: {
      vmSize: 'Standard_DS2_v2'
      osDiskSizeGB: 47
    }
    source: {
      type: 'PlatformImage'
      offer: 'windows-ent-cpc'
      publisher: 'MicrosoftWindowsDesktop'
      sku: 'win11-22h2-ent-cpc-m365'
      version: 'Latest'
    }
    customize: [{
      type: 'PowerShell'
      name: 'Install Choco and other tools'
      inline: settings[personaImage].inlineCommand
    }]
    distribute: [
      {
        type: 'SharedImage'
        galleryImageId: imageDefinition.id
        runOutputName: '${imageDefName}_Output'
        replicationRegions: array(location)
      }
    ]
  }
  dependsOn: [
    templateRoleAssignment
  ]
}
```

properties provides a quick hint on how to map them in your IaC.

- Target `type` specifies the distribution as a `SharedImage`.
- `galleryImageId` identifies the destination gallery for the shared image.
- `runOutputName` defines the name of the output, dynamically generated.
- `replicationRegions` lists the Azure regions where the image will be distributed.

A key highlight of assembling a custom image is defining the custom software and configurations that will be applied to a newly created VM image. The `customize` property maps directly to the command block, where these configurations are defined. Listing 5 shows Chocolatey being utilized to install the Java SDK and related tooling. Chocolatey is a popular open source package manager established for some years, although `WinGet` has recently gained significant traction. In this case, Chocolatey is installed with PowerShell by downloading the setup file, modifying the PowerShell execution policy, and running the `install.ps1` script provided by the Chocolatey distribution.

The `inlineCommand` property,

```
inline: ↲
  settings[personalImage].inlineCommand
```

specifies a series of commands executed directly on the VM during its setup. These commands are defined within the command block. PowerShell execution policy and software installation command steps are also included. In the example, the commands include the use of Chocolatey to install tools like Git, the Azure command-line interface (CLI), Visual Studio Code, and Java development resources. This relationship allows for customization of the VM image by installing the necessary tools and configurations automatically applied during provisioning.

Listing 5: Inline Command Settings

```
java: {
  publisher: 'MicrosoftWindowsDesktop'
  offer: 'windows-ent-cpc'
  sku: 'win11-22h2-ent-cpc-m365'
  inlineCommand: [
    'Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol
       = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.
       WebClient).DownloadString(\'https://community.chocolatey.org/install.ps1\'))'
    'choco install -y git'
    'choco install -y azure-cli'
    'choco install -y vscode'
    'choco install -y openjdk11'
    'choco install -y maven'
    '$vscode_extension_dir="C:/temp/extensions"; .. .. --install-extension vscjava.vscode-java-pack"
  ]
}
```

You have now seen the first three key artifacts to realizing a customized dev box: the dev box definition, the custom image by Bicep, and the inline command spec, which is your first chance to install customized software packages. Microsoft Dev Box offers additional facilities to customize YAML-based customization tasks, which I cover a bit later.

Building the Image

A command example briefly illustrates how Azure Image Builder automates the creation of customized VM images:

```
Invoke-AzResourceAction ↲
  -ResourceName "{imageTemplateName}" ↲
  -ResourceGroupName ↲
   "{resourceGroupName}" ↲
  -ResourceType "Microsoft.VirtualMachine↲
                Images/imageTemplates" ↲
  -ApiVersion "2024-02-01" ↲
  -Action Run -Force
```

The `Invoke-AzResourceAction` command executes the `Run` action, launching Azure Image Builder's pipeline. Customizations, like installing tools such as Chocolatey (introduced earlier), are applied during the image build process. This command triggers the image build pipeline and produces a custom VM image that can later be referenced for dev box purposes.

Taking a Test Drive

Now that you've been introduced to the technical details of Bicep-related DevCenter artifacts and resource definitions, it's time for a test drive.

To work with Bicep, which automatically translates the Bicep file into an ARM template in the background, you'll need to have `bicep.exe` installed and correctly pathed. The commands

```
az bicep install
az bicep version
```

install and validate that the Bicep command-line tool is available.

To begin, provision DevCenter resources by first creating a managed user identity or service principal on the Azure CLI with the command

```
az identity create
```

or use the equivalent PowerShell cmdlet. Capture the generated globally unique identifier (GUID), which is later inserted in the JSON settings. For this test drive, you'll use the Azure reference GitHub project *azure-quickstart-templates*, updating the `userPrincipalId` field in the `azure-deploy.parameters.JSON` file located in the `devbox-with-customized-image` subfolder. I have provided an additional GitHub project with the helper script `add-managed-user.ps1` to help automate the setup of a managed user identity [5].

To create the complete set of DevCenter resources in Azure, use the `New-AzResourceGroupDeployment` PowerShell cmdlet, which deploys all the resources from the template file to the specified resource group:

```
New-AzResourceGroupDeployment ↲
```

```
  -ResourceGroupName ↲
   "rg-cg-devcenter-3626" ↲
  -TemplateFile ".\main.bicep" ↲
  -TemplateParameterFile ↲
   "azuredeploy.parameters.JSON" ↲
  -suffix '64' -verbose -debug
```

`-TemplateFile` points to the Bicep template accessible from the local filesystem. In `-TemplateParameterFile`, you specify a local JSON file with the configuration settings injected during the deployment execution. The `-suffix` option adds a unique identifier to resource names to avoid conflicts. The `-verbose` flag generates detailed info during the deployment, and the `-debug` flag is added to dump additional details that might be useful for troubleshooting.

Before running this command or any preliminary commands, ensure that your Azure PowerShell context is properly initialized by the `Connect-AzAccount` cmdlet to authenticate and set up your session. Once you're ready to launch the resource deployment process, be prepared for a bit of a wait – deploying a resource group for a dev center design tailored to work with custom images can take more than 30 minutes to complete, so you have time to grab a coffee while `AzResourceGroupDeployment` triggers the heavy lifting of resource creation and image building. When it finishes, you see several lines of output printed to the console, including the deployment name, correlation ID, provisioning state, deployment debug log, and related Bicep template parameters.

I've created a PowerShell helper script, `deploy-devcenter-cust-image.ps1`, to simplify and streamline interacting with resource deployment command scenarios. You can access this script and other helpful resources from my GitHub repository [5].

Portal Landing

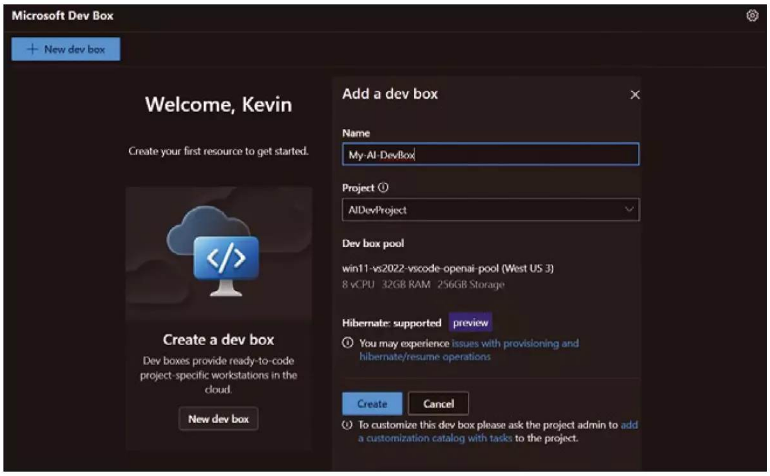
Once DevCenter and its companion resources have been deployed by ARM automation into your target Azure subscription, you can start creating virtual environments on the basis of the image templates associated with the selected dev box project. As shown in Figure 2, the Dev Box user experience starts at the Microsoft Dev Box website [6]. Alternatively, you can use the Microsoft sign-in page as another entry point. The web interface is designed to streamline the user self-service experience, providing only the selections necessary to round out the desired context of the dev box environment.

To get started, verify that the correct Azure roles (e.g., Dev Box Project Admin or Dev Box User) are assigned. If your dev box project isn't appearing in the dev box project selection list as expected, check Azure role assignments for the specific dev box project as a first step. Of note, multiple DevCenter resources can host projects and will render back into the global dev box web interface that is user context-specific.

When you're ready to create a dev box, start by selecting *New | New Dev Box*, and use the guided setup to define key parameters such as the project name, dev box pool, and a unique name for your dev box. The interface also displays useful information like pool limits, hibernation support, and customization options.

The Microsoft Dev Box portal web UI is the central hub of the self-service experience, making it easy for users to set up and manage their own dev boxes. With minimal setup and project-specific pool selections, the interface streamlines the process and lets users customize environments to fit their needs. Although the setup is simple, dev box creation can take 20 minutes to an hour, which is just enough time for you and your team to grab another coffee and snack.

Once your dev box environment is set up, you can access it with the standard Windows Remote Desktop Protocol (RDP) app or connect directly through a web browser. Figure 3 captures the web UI selection for the *Open in browser* pop-up


Microsoft Dev Box
+ New dev box
Welcome, Kevin
Create your first resource to get started.
Create a dev box
Dev boxes provide ready-to-code project-specific workstations in the cloud.
New dev box
Add a dev box
Create
Cancel

Name: My-AI-DevBox

Project ⓘ: AIDevProject

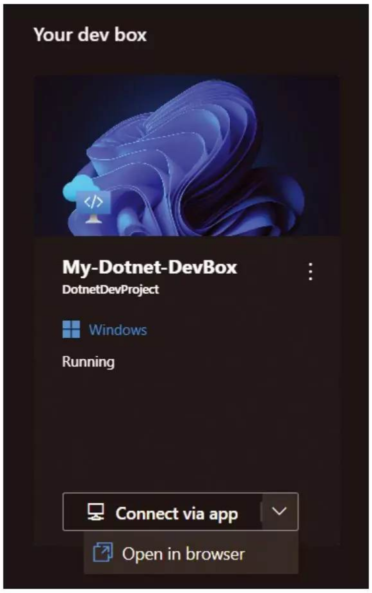
Dev box pool: win11-vs2022-vscode-openai-pool (West US 3)
8 vCPU 32GB RAM 256GB Storage

Hibernate: supported preview

ⓘ You may experience issues with provisioning and hibernate/resume operations

ⓘ To customize this dev box please ask the project admin to add a customization catalog with tasks to the project.

Figure 2: Create a dev box and add dev box project settings.



My-Dotnet-DevBox

DotnetDevProject

Windows

Running

Figure 3: Connect to the new DevBox instance in a browser-based session, and open in the browser (browser session, RDP).

menu item. Click this button to log in with your Azure AD credentials and proceed to the desktop. You will then be immediately presented with the In Session Settings dialog (Figure 4), which includes clipboard and file transfer. My best experience with these session settings or capabilities has been with Microsoft Edge. I had trouble making the file transfer work in Firefox.

Lifecycle Activities

Managing the lifecycle of a dev box environment involves setting up new projects, assigning user roles, assigning image galleries (e.g., shared by other DevOps teams, etc.), managing Dev Box-centric virtual environments, introducing new image templates, and so forth. These tasks can be realized at the command line with PowerShell cmdlets, the Azure `az` CLI, or both. For example, Listing 6 shows how to set up a new dev box project with essential details such as name and location. Assigning admins is another command task, for which you can use a PowerShell script to assign a DevCenter Project Admin role:

```
$resource = Get-AzResource ➘
  -ResourceGroupName ➘
   "rg-cg-devcenter-3626" ➘
  -ResourceType ➘
   "Microsoft.DevCenter/projects" ➘
  -ResourceName "project-65"
$resourceId = $resource.ResourceId
New-AzRoleAssignment ➘
  -ObjectId (Get-AzADUser ➘
   -UserPrincipalName ➘
    "aduser@domain.com").Id ➘
  -RoleDefinitionName ➘
   "DevCenter Project Admin" ➘
  -Scope $resourceId
```

Dev Box virtual environments consume Azure compute power, memory, and other resources, affecting Azure cloud costs. Monitoring and managing these resources are key to managing cloud expenditures. The Azure CLI offers a straightforward way to manage dev boxes. For instance, to stop a running dev box, use the `az devcenter` command syntax:

```
az devcenter dev dev-box stop ➘
  --dev-center "devcenter-65" ➘
  --project "DotnetDevProject" ➘
  --dev-box-name "My-Dotnet-DevBox"
```

You can use a variety of `az devcenter` command verbs; check the Microsoft Learn documentation for more information [7].

It's best to verify that the `devcenter` extension is installed locally before running these Azure CLI commands and, if not listed, install it:

```
az extension list
az extension add --name devcenter
```

Once installed, the Azure CLI provides additional `devcenter` commands as a comprehensive and efficient management toolkit to display a dev box status, restart it, or bring it back online.

Integrating newly introduced image galleries into your dev center is an essential step in the lifecycle management of virtual environments that allows you to access and use updated images easily for your dev boxes. You can attach a gallery to your dev center with the PowerShell cmdlet:

```
New-AzDevCenterAdminGallery ➘
  -Name "NewGallery-65B" ➘
  -DevCenterName "devcenter-65" ➘
  -ResourceGroupName ➘
   "rg-cg-devcenter-3626" ➘
  -GalleryResourceId ➘
   "/subscriptions/.. c76f/resourceGroups/➘
    rg-cg-devcenter-q-3626/resourceGroups/➘
    rg-cg-devcenter-q-3626/providers/➘
    Microsoft.Compute/galleries/gallery65"
```

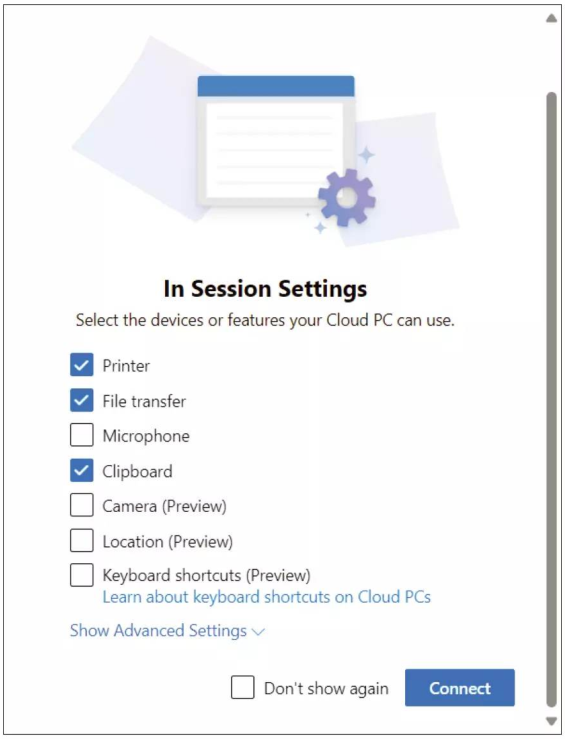
This command links the specified image gallery to your dev center, ensuring that users have streamlined access to the latest VM images tailored to their development needs.

Further Customization with Tasks

The Dev Box `tasks` is a new capability (under preview) that supports the creation of a YAML-defined action to configure a dev box (e.g., by installing software with PowerShell scripts and `WinGet`). Tasks reside in a Git-based catalog that you attach to a dev center for easy DevCenter project sharing. The Dev Box user can optionally apply these tasks by

Listing 6: New Dev Box Project

```
New-AzDevCenterAdminProject -ResourceGroupName "rg-cg-devcenter-3626" -DevCenterId
  "/subscriptions/.. c76f/resourceGroups/rg-cg-devcenter-q-3626/providers/
  Microsoft.DevCenter/devcenters/devcenter-65" -Name "project-65" -Description
  "Project for .NET Development with internal networking" -Location "westus3"
```



In Session Settings

Select the devices or features your Cloud PC can use.

☑ Printer
☑ File transfer
☐ Microphone
☑ Clipboard
☐ Camera (Preview)
☐ Location (Preview)
☐ Keyboard shortcuts (Preview)
Learn about keyboard shortcuts on Cloud PCs

Figure 4: Dev box In Session Settings dialog.

checking the *Apply customizations* option when adding a new dev box environment or instance.

The following code fragment demonstrates a configuration-as-code YAML task customization that installs Wireshark into a dev box environment by `WinGet` package management:

```
$schema: 1.0
tasks:
 - name: winget
   parameters:
     package: WiresharkFoundation.Wireshark
     runAsUser: true
```

Microsoft continues to enhance the Dev Box offering in Azure, with regular updates to APIs and Bicep definitions in preview releases. During a proof of concept with Azure Virtual Desktop (AVD), the Microsoft account team recommended Dev Box as a better fit for my needs, which involved providing cloud-based access to a complex development toolchain focused on delivering business value. If you have a possible use case in mind, instead of proposing a classical virtual desktop interface (VDI) solution, you might instead evaluate Dev Box. Check it out! ■

Info

[1] Microsoft Dev Box overview: [https://learn.microsoft.com/en-us/azure/dev-box/overview-what-is-microsoft-dev-box]

[2] Microsoft Dev Box quick start: [https://learn.microsoft.com/en-us/azure/dev-box/quickstart-configure-dev-box-service]

[3] Azure quick start template – devcenter: [https://github.com/Azure/azure-quickstart-templates/tree/master/quickstarts/microsoft.devcenter]

[4] Azure IaC templates – devcenter: [https://learn.microsoft.com/en-us/azure/templates/microsoft.devcenter/devcenters]

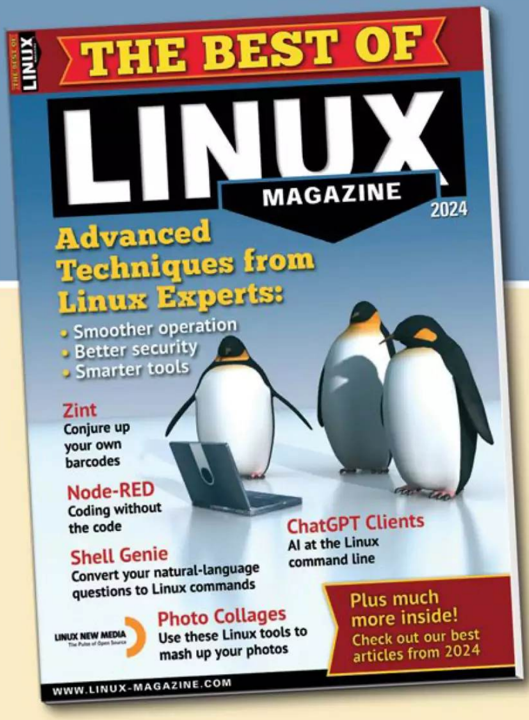
[5] Article scripts and other source code artifacts: [https://github.com/krwittmer/admin-magazine-source.git]

[6] Microsoft Dev Box: [https://devbox.microsoft.com]

[7] Azure devcenter command-line support: [https://learn.microsoft.com/en-us/cli/azure/devcenter/dev/dev-box?view=azure-cli-latest]

Author

Kevin Wittmer is a chief IT technologist and enterprise architect living in the Chicago area. When Kevin's not immersed in AI, cloud, Linux, SRE, or playing around with fractal generation in Python, you can find him on the tennis court. To reach out to Kevin, contact him via LinkedIn at [www.linkedin.com/in/kevininchicago].


THE BEST OF
LINUX MAGAZINE
2024
Advanced Techniques from Linux Experts:
Smoother operation
Better security
Smarter tools
Zint Conjure up your own barcodes
Node-RED Coding without the code
ChatGPT Clients AI at the Linux command line
Shell Genie Convert your natural-language questions to Linux commands
Photo Collages Use these Linux tools to mash up your photos
Plus much more inside! Check out our best articles from 2024
WWW.LINUX-MAGAZINE.COM

THE BEST OF
LINUX MAGAZINE 2024

Advanced Techniques from Linux Experts

This new special edition brings you the best and most practical articles from the 2024 editions of *Linux Magazine*.

Whether you're new to Linux or a seasoned veteran, you'll find something useful in the tips, tools, and technologies inside.

ORDER ONLINE:
shop.linuxnewmedia.com

Lean Linux distribution for Kubernetes

Featherlight

In the world of container virtualization, the operating systems of compute nodes are largely degraded to non-player characters that can do little more than start and stop containers. Talos Linux takes the game to the extreme and offers a system for Kubernetes that weighs in at less than 90MB.

By Martin Loschwitz

Huge environments for automation and orchestration used to be standard – think Ansible, Puppet, and others; however, today's buzzword is "reduction." Logically, a system that only needs to run containers can easily do without a massive user space; you don't need additional services on the system or extravagant configuration options. Strictly speaking, you don't even need a package manager – at least if you don't intend to update the system while it is running. Instead, you would remove all the containers and then reinstall more up-to-date software. A simple Linux kernel with a paired down set of drivers in combination with a runtime environment for containers – typically Docker Community Edition (CE) or Podman – is absolutely fine for this scenario. Proponents of immutable IT have developed and radicalized their strategy. Whereas some Linux distributions still offer SSH and a local shell, immutable IT approaches assume you don't even need SSH to log in to remote systems. One of the solutions that takes this approach is Talos Linux, a distribution for operating systems whose core task is being part of a Kubernetes cluster.

Reduction

Reduction is more than an academic exercise in the frugal use of space on storage devices. A great deal of work and maintenance overhead is offloaded from the Linux distribution vendor. However, it also shifts a significant part of the support overhead from the Linux providers to the application vendor, because Red Hat, SUSE, and the like can correctly assert that users are getting MySQL in a container directly from Oracle. According to the Linux distributors' common understanding, any problems that arise as a result need to be settled between the program vendor and the customer, and not between the customer and the distributor. In short, as Linux providers see it, this principle offers so many benefits that it seems irresistible.

Immutable IT

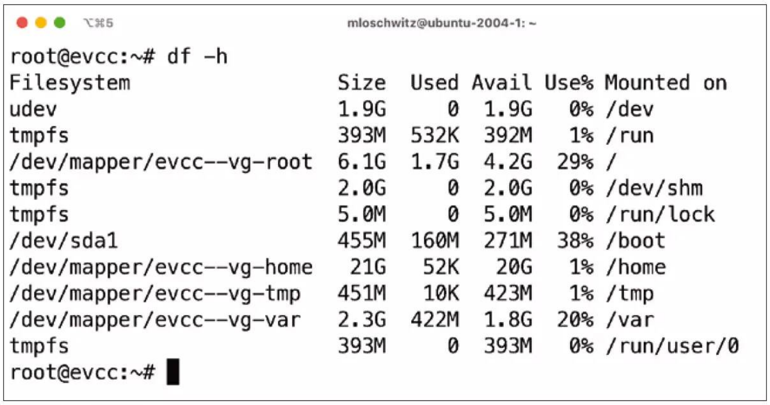
If you want to understand the functional principle of Talos, you first need to take a closer look at what immutable IT means. "Immutable" is just a fancy way of saying unchangeable. The basic idea behind an immutable infrastructure or immutable IT is to keep the number of variable components on the individual systems as low as possible. This principle is now also being applied to other services in the IT sector, including storage. Immutable storage translates write once, read many (WORM) storage; in other words, once the data has been written to the data carrier, no edits are permitted. Elements of this approach can also be found in immutable infrastructure solutions. The aim is for the data center's internal automation system to roll out a single system once, then provision it with a complete configuration in a standard way and use the system without any changes for the rest of its service life.

Precursors of this strategy were already making the rounds back in 2014. Kristian Köhntopp, a veteran of the German IT scene, was already making guest appearances at conferences at the time, saying that you needed to reinstall every system after an administrator had logged into it over SSH, because it could no longer be assumed with absolute certainty that the system complied with all the applicable security and compliance regulations, and the overhead required to restore this status manually was greater than that for reinstalling the system. Köhntopp also pointed out that admins needed to create an internal troubleshooting ticket for every required SSH login on a system because either the automation was not good enough (it failed to field a technical problem) or the centralized logging and metrics data acquisition was not good enough (it was not possible to prevent the issue without logging in to the system). Back then, when private clouds were just gaining speed, Köhntopp's approach was decried in many places as too radical and impractical.

Ten years later, projects like Talos just go to show that Köhntopp was right. However, don't ignore Kubernetes development as part of the equation. Kubernetes basically sees itself as a kind of data center in a box. The declared intent is to automate the vast majority of tasks that occur in the data center on a daily basis and give admins control over them by API, which includes creating new instances, terminating running instances, adding and removing individual network configurations or services, and many other procedures. Kubernetes has become so complex over the years that it provides most of the functionality required for a data center in a box and has done so for a long time.

Lead Image © joingate, 123RF.com

Red Hat has been working on a core distribution for some time, and it has already become a reality to some extent in Red Hat Enterprise Linux (RHEL) 9. SUSE is taking a similar approach with its Adaptable Linux Platform (ALP), and Canonical is planning a miniaturized Ubuntu, as well. Although still hesitant, Debian sooner or later is also likely to develop a microdistribution that offers very little more than a kernel and Podman or Docker.


⌥⌘5
mloschwitz@ubuntu-2004-1: ~

```
root@evcc:~# df -h
Filesystem                   Size  Used Avail Use% Mounted on
udev                         1.9G     0  1.9G   0% /dev
tmpfs                        393M  532K  392M   1% /run
/dev/mapper/evcc--vg-root    6.1G  1.7G  4.2G  29% /
tmpfs                        2.0G     0  2.0G   0% /dev/shm
tmpfs                        5.0M     0  5.0M   0% /run/lock
/dev/sda1                    455M  160M  271M  38% /boot
/dev/mapper/evcc--vg-home     21G   52K   20G   1% /home
/dev/mapper/evcc--vg-tmp     451M   10K  423M   1% /tmp
/dev/mapper/evcc--vg-var     2.3G  422M  1.8G  20% /var
tmpfs                        393M     0  393M   0% /run/user/0
root@evcc:~# ▌
```

Figure 1: Even a minimal Ubuntu installation occupies more than 2GB on disk when unpacked. Talos Linux makes do with less than one twentieth of this.

Radical Approach

Strictly speaking, even distributions such as RHEL 9 and SUSE ALP are already lagging behind, although they are not even officially available yet (in ALP's case). Whereas ALP or RHEL still offer SSH and a local shell, immutable IT approaches like Talos Linux assume you don't even need an option to log in to remote systems (see the "Immutable IT" box). Talos Linux occupies less than 90MB when installed, compared with Ubuntu, which hogs almost 2GB in a minimal installation (Figure 1).

Also implied is that more and more functionality is migrating from the nodes' infrastructure to the Kubernetes level and that the system's software can become dumber in return. Talos is a very concrete implementation of this principle. Once the distribution has been installed on a system, you should be able to control the components that make up the system remotely by API to influence the configuration, with the system itself barely visible after the install. Instead, Kubernetes centrally controls and handles all the pending tasks for the system.

Talos does not come with SSH, and it does not give you a login shell; instead, it starts containers. Talos Linux can be controlled remotely by API; for example, you can configure where to send the logfiles. Talos does not try to be a general purpose distribution. It is a tool that specifically targets administrators of Kubernetes clusters. Thus far, Kubernetes cluster admins have faced the challenge that – in a correctly configured cluster – Kubernetes makes up the lion's share of the configuration with its Kubelets on the target systems. Nevertheless, if you want to use systems as compute nodes in a Kubernetes setup, they still need a basic Linux-based installation, which is reason enough to take a closer look:

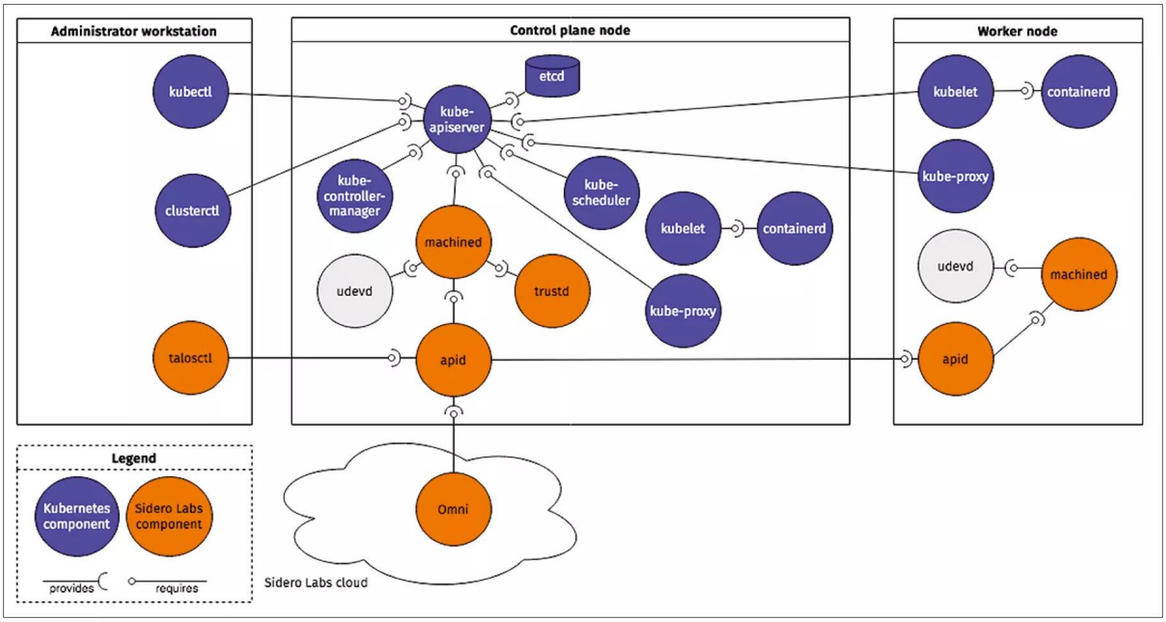

Administrator workstation
kubectl
clusterctl
talosctl
Control plane node
etcd
kube-apiserver
kube-controller-manager
kube-scheduler
kubelet
containerd
machined
udevd
trustd
kube-proxy
apid
Worker node
kubelet
containerd
kube-proxy
udevd
machined
apid
Omni
Sidero Labs cloud
Legend
Kubernetes component
Sidero Labs component
provides
requires

Figure 2: To lose weight, Talos Linux has dumped overboard most of the standard components that modern Linux distributions offer and replaced them with its own components. © Talos Linux

What is it that motivates the Talos developers? What technical strategies does Talos use? What do you need to consider if you decide to use Talos?

What Talos Comprises

Suspicious admins question what they are actually dealing with when they come face to face with Talos Linux and what components the 80MB image comprises (Figure 2). Although Talos Linux has a Linux kernel at its core, the developers of the distribution have surrounded it with a kind of protective shield, because the kernel is configured and built in line with the specifications of the Kernel Self Protection Project (KSPP) [1].

Technically speaking, KSPP is very interesting. Its creators assume that the Linux kernel always has (critical) errors and it will take a long time to discover their existence, which is why they advocate making the Linux kernel more robust in its handling of security issues by defining special configurations and introducing a variety of functions. Even if security problems exist, it should not cause exploitable vulnerabilities, say the people at KSPP. They help you out with a number of tools in this respect (e.g., a list of kernel settings with recommendations from the KSPP project).

Frugality is the most important basic rule: Drivers that are not on the system cannot be used as attack vectors. Because it is built in line with KSPP rules, the Talos kernel is lean. Although it includes most drivers you need for state-of-the-art hardware, you should not expect bells and whistles. And why would you want them? You have no way to log in to a Talos Linux system over SSH anyway, and reloading kernel drivers is not intended – especially not if they can't even be used meaningfully in a Kubernetes context.

From the administrator's point of view, the other core components belonging to Talos are far more interesting than the operating system kernel anyway. The central components in Talos Linux – and that means everything that keeps the system running – are genuine Talos in-house developments. This arrangement is evident because Talos contains practically none of the familiar system services that other distributions use, such as systemd. Everything that Talos contains under the hood is based on Go and comes from Sidero Labs, the company behind Talos Linux.

For example, `apid` is an integral part of every Talos installation, providing a gRPC (remote procedure call, originally created by Google) endpoint for each individual Talos node that lets you import a node-specific configuration. On the userspace side, you use a counterpart named `talosctl` to communicate directly with `apid` on each Talos node and pass commands to the individual daemon instances.

No Trace of systemd

As you know, the Linux kernel starts the init process, to which it then hands over control of the entire system. Practically all of today's distributions use systemd as the init. RHEL 9 or SUSE ALP also include systemd, presumably because both distributors put a substantial amount of development work into the init system. At the same time, systemd has often been a bone of contention within the Linux community because of its all-in way of taking control of system processes. This makes it almost revolutionary on the one hand, yet understandable on the other, that Talos Linux does not hold systemd in high regard, which prompted the people at Talos simply to replace the component with its own variant of the init process.

The Talos-style init is called `machined`; it is passed into the kernel as the parameter of `init=`. Of course, `machined` does not offer anything like as many options as systemd; instead, the focus is on starting all services needed for Talos operation (e.g., `apid`) and on guaranteeing the functionality of the Open Container Initiative (OCI)-compatible runtime environment for containers, which Talos includes in the form of containerd, a well-established and well-known runtime used on other distributions with close ties to Kubernetes. In Talos, `machined` is also supported by `udevd`, although it is not the same animal as in the systemd universe. Instead, the Talos developers forked `eudev`, the daemon responsible for managing the devices in the `dev/` directory on Gentoo Linux.

Talos also rolls out for all nodes in an installation a component named `trustd` that acts as Kubernetes control nodes, establishing a public key infrastructure (PKI) between the Talos nodes and ensuring that the individual systems trust each other, so that they can, say, use `apid` to exchange commands.

That sort of winds up the Talos Linux component charts. You do not need more than the five or six services just described to access the entire Talos feature set.

Talos Workflow

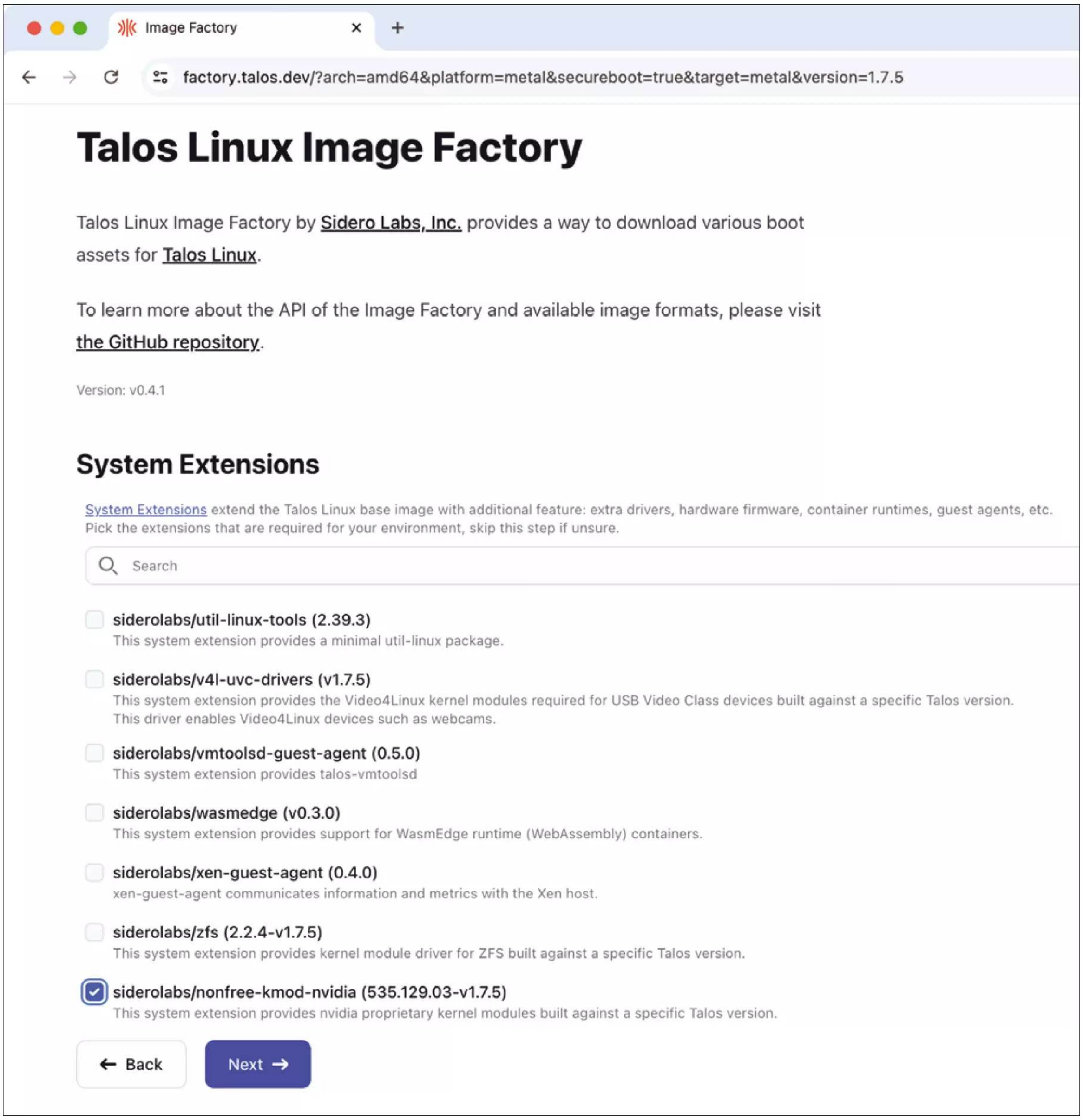
Hard-working Linux admins who square up to Talos for the first time might feel they are in a science fiction movie. Many familiar processes and procedures work in a fundamentally different way than on legacy Linux distributions, starting with how you install the system. Although the vendors do offer ISO images, they obviously prefer a Trivial File Transfer Protocol (TFTP)-based and preboot execution environment (PXE)-based approach over the network, which means you will need a couple of infrastructure services. A TFTP server and a DHCP server are not enough to get the job done. For practical reasons, you will also need for each system that will be using Talos Linux a network time server and a separate configuration file that stores all the central configuration settings for the target machine. The Talos Linux claim of absolute control over the environment goes far beyond what administrators tend to dislike about systemd.

The method to this madness involves PXE, which can be set up relatively quickly and used with generic images that come directly from the vendor. Because local changes on the respective machines are not intended anyway, a configuration file residing centrally on a server for an instance can store all the required parameters. The `talosctl` tool lets you restart a machine or line it up for a reinstall by PXE (e.g., if something went wrong on the system itself).

Once Talos has been rolled out, the system automatically configures the remaining necessary services. If it is a controller node for Kubernetes, Talos automatically rolls out all the Kubernetes components on the system, but in the form of containers. The system leaves you largely free to choose which components to use, such as the Container Network Interface (CNI) plugin for Kubernetes. You also configure the individual Talos instances at runtime, again with `talosctl`, which is then used to import and enable node-specific configuration files. In contrast, practically the entire filesystem on Talos is immutable at runtime.

Your Image Factory

Incidentally, Sidero Labs makes life surprisingly easy for Talos newcomers looking to migrate. The provider offers the free Talos Linux Image Factory (**Figure 3**) online; you can use it to create customized ISO images, PXE boot images, images for operation in the cloud, and images for various other purposes. Once again, the Talos "frugal first" doctrine applies. Whereas other manufacturers



factory.talos.dev/?arch=amd64&platform=metal&secureboot=true&target=metal&version=1.7.5

Talos Linux Image Factory

Talos Linux Image Factory by Sidero Labs, Inc. provides a way to download various boot assets for Talos Linux.

To learn more about the API of the Image Factory and available image formats, please visit the GitHub repository.

Version: v0.4.1

System Extensions

System Extensions extend the Talos Linux base image with additional feature: extra drivers, hardware firmware, container runtimes, guest agents, etc. Pick the extensions that are required for your environment, skip this step if unsure.

☐ siderolabs/util-linux-tools (2.39.3)
This system extension provides a minimal util-linux package.

☐ siderolabs/v4l-uvc-drivers (v1.7.5)
This system extension provides the Video4Linux kernel modules required for USB Video Class devices built against a specific Talos version. This driver enables Video4Linux devices such as webcams.

☐ siderolabs/vmtoolsd-guest-agent (0.5.0)
This system extension provides talos-vmtoolsd

☐ siderolabs/wasmedge (v0.3.0)
This system extension provides support for WasmEdge runtime (WebAssembly) containers.

☐ siderolabs/xen-guest-agent (0.4.0)
xen-guest-agent communicates information and metrics with the Xen host.

☐ siderolabs/zfs (2.2.4-v1.7.5)
This system extension provides kernel module driver for ZFS built against a specific Talos version.

☑ siderolabs/nonfree-kmod-nvidia (535.129.03-v1.7.5)
This system extension provides nvidia proprietary kernel modules built against a specific Talos version.

Figure 3: **The Talos Image Factory lets you build customized images of the distribution, which can also contain additional components such as the commercial NVIDIA drivers.**

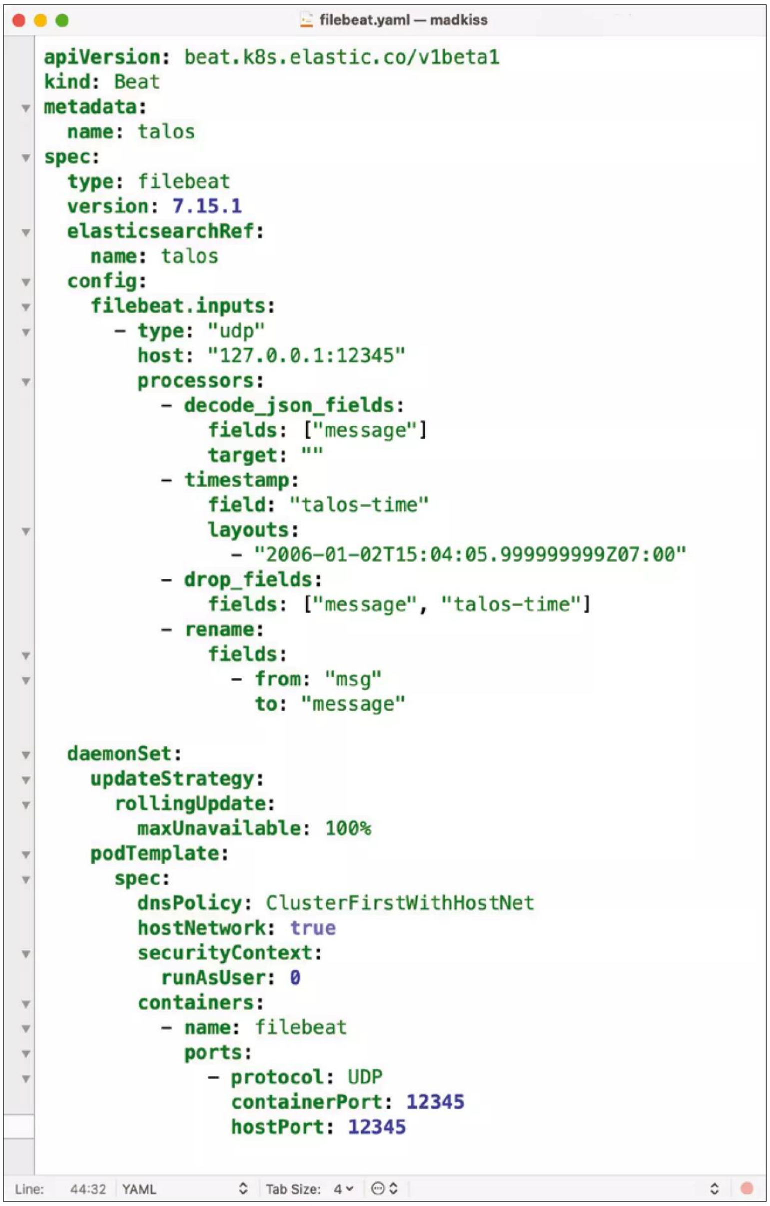
equip their distribution kernels with a plethora of drivers enabling the use of services such as distributed replicated block devices (DRBDs) or NVIDIA GPUs, these drivers are missing from Talos standard images.

If something you need is missing, you can turn to the Image Factory on the Talos website [2] to piece together a suitable image yourself. Secure Boot support is also missing from the standard images but can be added at the push of a button. On top of this, ready-made Talos images are available for use on most single-board computers, such as the Raspberry Pi. Talos is also easy to use in the context of a home lab and is highly recommended for this purpose.

Simple Tasks, Difficult to Do

Anyone looking at Talos for the first time is likely to laud the clarity and streamlining of its architecture. An inherently immutable system that can be controlled and fully monitored from the outside by the API tends to make you want more, especially given that the entire configuration for Talos can be stored as infrastructure-as-code and thus managed on GitHub.

However, the inability to log in quickly to a system and track the environment can be a pain in the neck if you experience difficulties. If you ever need to plow through the Talos documentation to find out how centralized logging and centralized metrics data acquisition are intended to work, you will soon understand what I mean.



```
apiVersion: beat.k8s.elastic.co/v1beta1
kind: Beat
metadata:
  name: talos
spec:
  type: filebeat
  version: 7.15.1
  elasticsearchRef:
    name: talos
  config:
    filebeat.inputs:
      - type: "udp"
        host: "127.0.0.1:12345"
        processors:
          - decode_json_fields:
              fields: ["message"]
              target: ""
          - timestamp:
              field: "talos-time"
              layouts:
                - "2006-01-02T15:04:05.999999999Z07:00"
          - drop_fields:
              fields: ["message", "talos-time"]
          - rename:
              fields:
                - from: "msg"
                  to: "message"

  daemonSet:
    updateStrategy:
      rollingUpdate:
        maxUnavailable: 100%
    podTemplate:
      spec:
        dnsPolicy: ClusterFirstWithHostNet
        hostNetwork: true
        securityContext:
          runAsUser: 0
        containers:
          - name: filebeat
            ports:
              - protocol: UDP
                containerPort: 12345
                hostPort: 12345
```

Figure 4: It is something of a Sisyphean task to extract logfiles from Talos in a meaningful way. This Filebeat example shows a potential solution.

Looking into logging issues is almost like looking into a black hole. Talos distinguishes between logfiles belonging to its own services, kernel logs, and the logs that additional services in containers generate. You can configure a log target for Talos itself in the Talos configuration of the node, and Talos will happily send the logfiles in JSON format over UDP or TCP. You could then use Grafana to work with the files, and the kernel logs can be exported elsewhere in this way as JSON.

Everything becomes far trickier wherever you need to route additional logfiles from Kubernetes services (e.g., those from Kubelets) to a logging host. Talos relies on Filebeat (Figure 4), Fluentd, or Vector, but all of these tools first need to be set up with separate Talos configurations.

You can at least specify the path on the target system from which you want to pick up the desired logs in `/var/log`. However, if the various logging mechanisms do not offer a way of extracting the information from the system, it is simply not accessible in any way. As long as everything works as planned, it doesn't matter, but if malfunctions or difficulties occur, direct debugging is impossible.

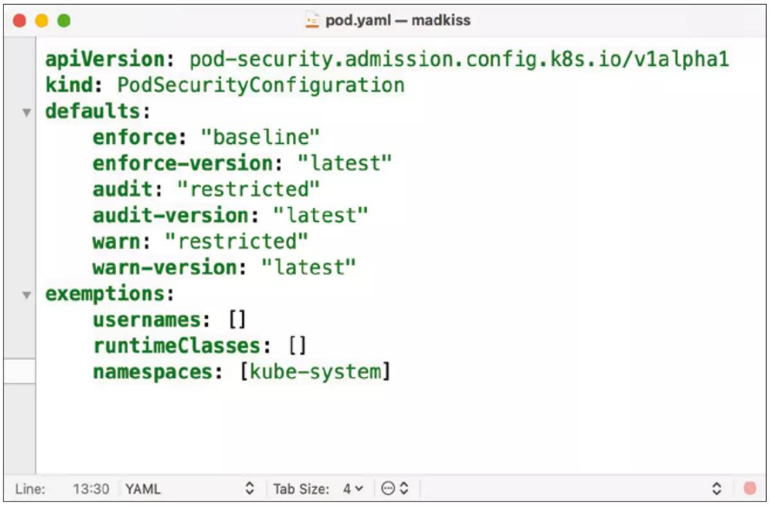
Additionally, adding Talos instances to monitoring systems such as Prometheus is not convenient. The Prometheus Node Exporter, for example, which collects central metrics from the physical system, first needs be laboriously integrated into the setup manually with `talosctl`. An upgrade to the Talos security framework is typically needed, as well; otherwise, the exporter will not be able to access the hardware components you want to monitor (**Figure 5**). Generally speaking, you can tell that observability is not a major focus in the development of Talos, so it has room for improvement in some areas. One thing noticeably dampened my mood with regard to what is otherwise a totally cool product: Extracting required information from Talos systems can be very complex, not least because you don't have any options for direct access.

Mixed Feelings

If you are interested in Talos Linux, you first need to think about its designed use and the designated target group. Talos clearly seeks to be the dream distribution for operating systems whose core task is being part of a Kubernetes cluster. For this very clearly delineated use case, most of the standard components that you would otherwise find on a Linux system are simply not necessary. As such, it is only logical for Talos to leave them out. You can quickly drop the idea of using Talos as a Linux distribution for tasks outside of the Kubernetes context. At best, you might be able to develop a separate distribution as a Talos fork that defines different priorities and excels in other ways. Without doubt, Talos is not suitable as a general purpose distribution.

Moreover, the Talos developers might need to think about whether Talos overdoes the idea of abstraction in some places and simply makes things too complicated. The fact that a Talos node's network configuration can be changed remotely with `talosctl` is technically cool on the one hand, but experienced observers might wonder why the people at Talos think that this procedure is superior to using Ansible to edit a Network Manager file. The fact that SSH is missing also forces you to use Talos's designated methods to handle tasks. As a result, configuring monitoring, alerting, trending, and centralized logging can quickly become a tedious chore.

On the upside, you can look forward to a Linux distribution that has been reduced to the max and whose attack vector is likely to be significantly smaller even than that of Linux distributions that claim to be particularly secure. If Kubernetes is just popping up on your radar screen and you are looking to explore, you will want to take Talos Linux for a trial run and carry out some extensive testing. Just remember that some frustration is inevitable, especially at the beginning of your evaluation. ■



```
apiVersion: pod-security.admission.config.k8s.io/v1alpha1
kind: PodSecurityConfiguration
defaults:
    enforce: "baseline"
    enforce-version: "latest"
    audit: "restricted"
    audit-version: "latest"
    warn: "restricted"
    warn-version: "latest"
exemptions:
    usernames: []
    runtimeClasses: []
    namespaces: [kube-system]
```

Figure 5: Rolling out Prometheus to Talos has no simple solution; to do so, you painstakingly have to manipulate the default pod definitions first, because the node exporter will otherwise not start.

Info

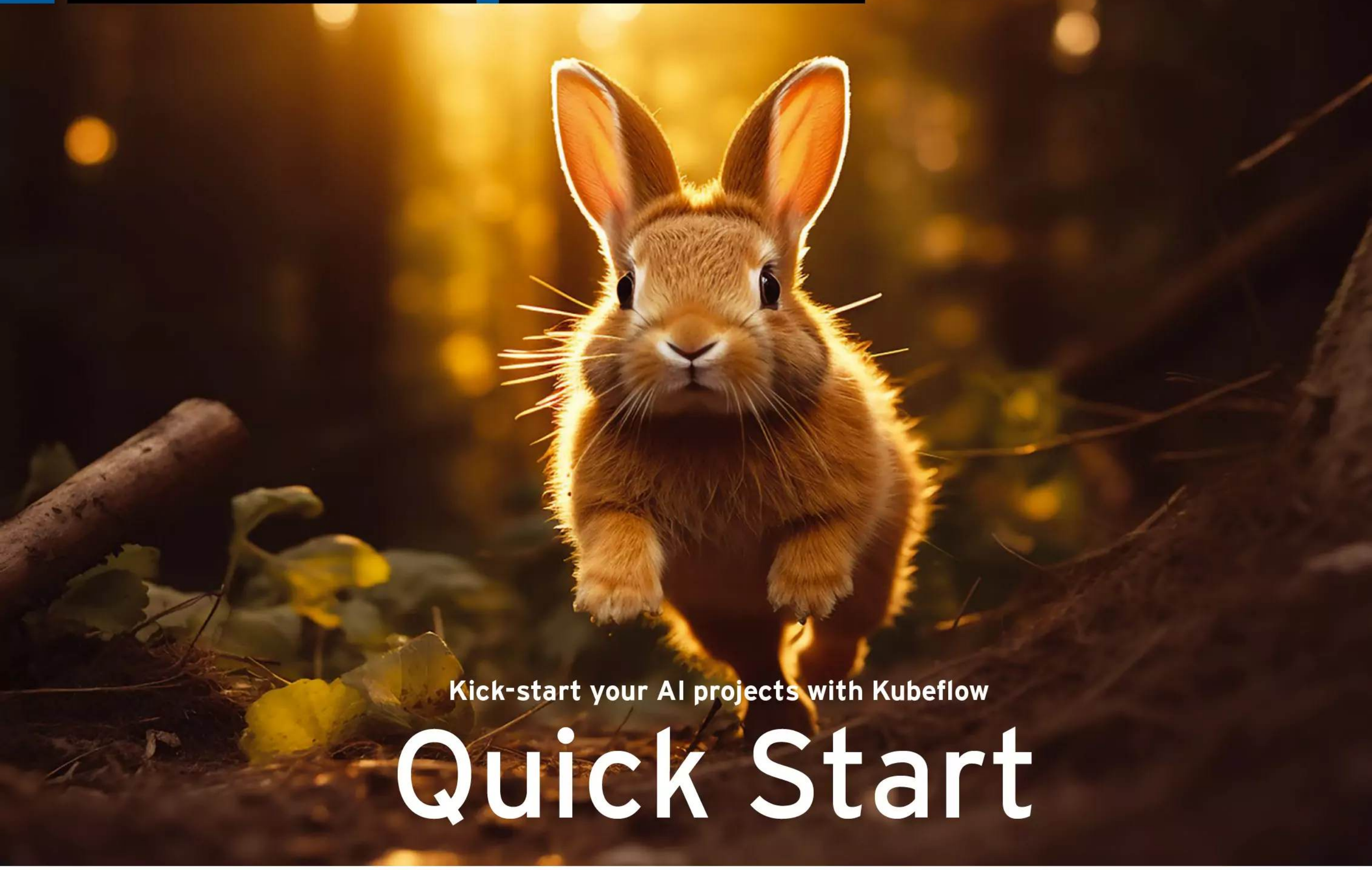
[1] KSPP: [https://github.com/KSPP/kspp.github.io/blob/main/index.md]

[2] Image Factory: [https://factory.talos.dev]

The Author

Martin Loschwitz is the founder and managing director of True West IT Services GmbH, which offers scalable IT infrastructure based on OpenStack and Kubernetes.

■



Kick-start your AI projects with Kubeflow

Quick Start

Training language models and AI algorithms requires a powerful infrastructure that is difficult to create manually. Although Kubeflow promises a remedy, it is itself a complex monster … unless you are familiar with the right approach that lets you get it up and running fairly quickly. By Martin Loschwitz

Artificial intelligence (AI) and machine learning have been on everyone's lips for some time now. The more accessible the technology becomes, the more developers will realize they need to develop algorithms that pervade the everyday lives of the masses and make life easier. However, interested developers initially face a challenge that has nothing to do with AI and language models: providing the technical infrastructure.

Kubeflow [1] is an open source machine learning platform that promises to offer all the tools you need for AI development, which means the components number well over 70, and they need to be rolled out and operated with the right configurations and in a correct and coordinated sequence. The enormity of this undertaking explains why more than a few developers feel they have been taken for a ride. Anyone who has never looked at Kubeflow will need more or less as much time to familiarize themselves with it as they would take to organize the required components themselves … and I've not even mentioned Kubernetes itself (see the "Taming the AI Infrastructure" box). Any administrator who has ever rolled out K8s knows that success is by no means guaranteed.

In this article, I provide a hands-on guide on how to get Kubeflow up and running quickly with a freshly created AWS account. The installation uses Cognito for user management but does not connect to external components such as Amazon Simple Storage Service (S3), instead using the components already available in Kubeflow.

Kubeflow

Kubeflow is surfing the popular wave surrounding Linux containers (Figure 2). As the name suggests, the platform is based on Kubernetes, and the "flow" in the name at least indicates that it is about providing workflows. Kubeflow is particularly interesting from an AI perspective, in that the workflows and tools provided by the environment almost exclusively relate to AI workloads and the training of language models.

Kubeflow provides developers with all the tools and integration components they need to start working on AI models right away without having to worry about the underlying infrastructure. There is a catch, of course: Kubeflow is a monster. The Kubeflow pipelines (KFP) platform comprises a number of components that need to be rolled out in Kubernetes, just like a bone fide microarchitecture application. Besides legacy services such as etcd and Istio, you need tools such as a dedicated DNS service or a registry for container images.

Hyperscalers to the Rescue

Getting Kubernetes and Kubeflow to work together on your hardware turns out to be a laborious undertaking, not least because Kubeflow identifies more as a kind of software collection that can be rolled out and used in a variety of ways, but which end users

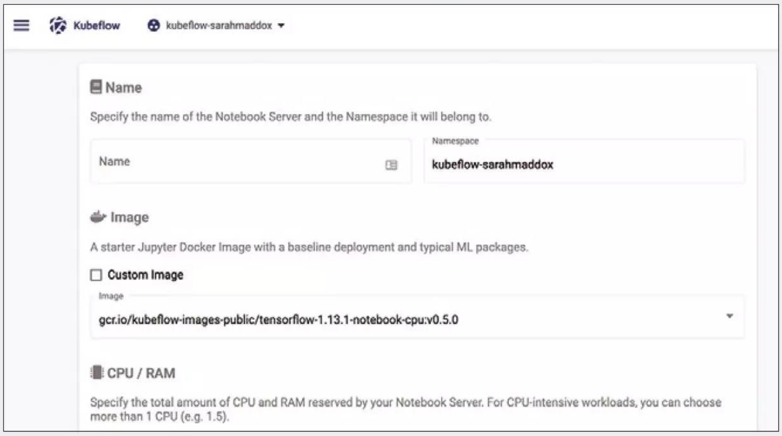
Lead Image © austler, 123RF.com

Taming the AI Infrastructure

Tools for building and training AI-based language models have been available for a long time. They often originate from the open source environment, which benefits from its proximity to academic circles and science when it comes to AI. A large number of the language models available today use open source scripting languages and libraries and are themselves published under free licenses.

In the context of machine-based learning, providing the technical infrastructure is no mean task. In fact, every AI model behaves like a complete program, to which components such as the material the model needs for training are added. The AI community has long since established methods and rules for developing compliant models and making them available to others.

The processes are similar to those in traditional software development. Continuous development and continuous delivery (CI/CD) play just as important a role as do APIs, which expose models to the outside world by standardized paths and, in turn, implies the use of pipelines (i.e., defined processes containing many steps), with the help of which a model can be developed and trained from the first evolutionary stage to completion. Git also plays a crucial role in version management. If AI models are available in program form, they can also be managed, edited, updated, and ultimately executed like any other program, which means a huge amount of prep work for developers who just want to experiment with and initiate research into AI. As ever, the open source community knows exactly what to do. The fact that so many students have shifted their focus from traditional programming topics to state-of-the-art tasks such as AI and language models has prompted people to think about environments that are designed to make it easier to get started in AI development. Jupyter is an excellent, but unfortunately commercial, example, as are solutions such as TensorFlow (Figure 1).


Kubeflow
kubeflow-sarahmaddox

Name

Specify the name of the Notebook Server and the Namespace it will belong to.

Name

Namespace: kubeflow-sarahmaddox

Image

A starter Jupyter Docker Image with a baseline deployment and typical ML packages.

☐ Custom Image

Image: gcr.io/kubeflow-images-public/tensorflow-1.13.1-notebook-cpu:v0.5.0

CPU / RAM

Specify the total amount of CPU and RAM reserved by your Notebook Server. For CPU-intensive workloads, you can choose more than 1 CPU (e.g. 1.5).

Figure 1: Kubeflow's notebook component offers seamless integration into various AI frameworks, such as Jupyter and TensorFlow. Kubeflow also implements the TFServe feature for Tensorflow as a separate component. © Kubeflow

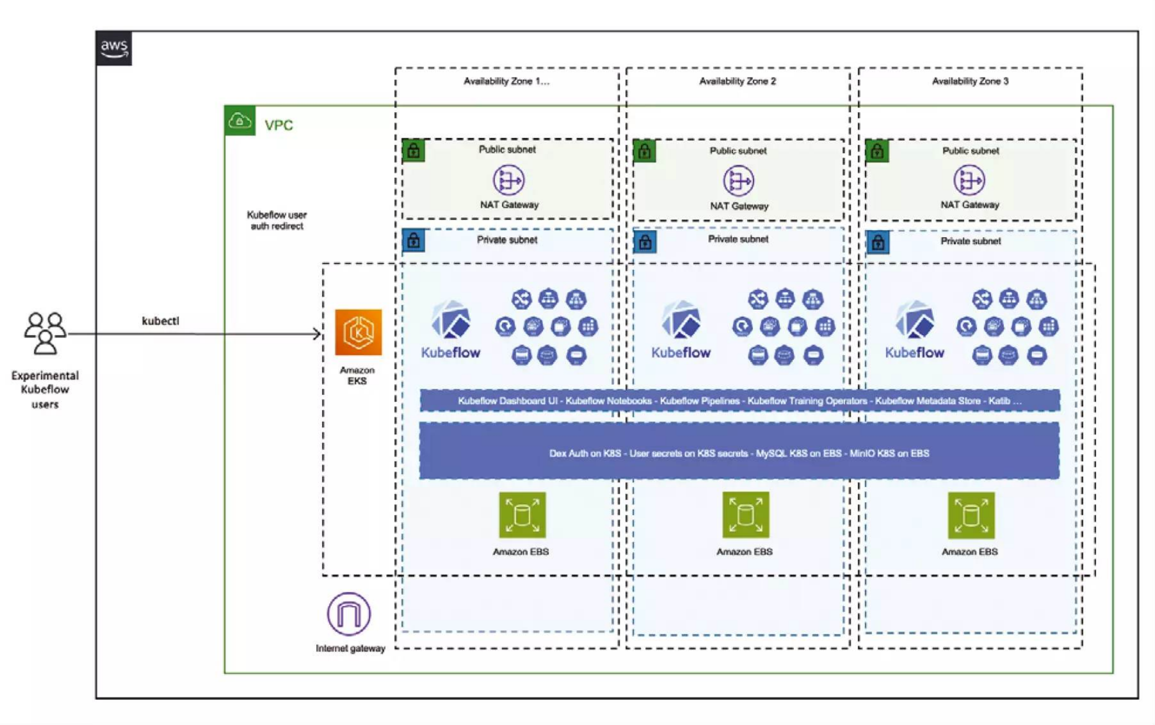

aws
VPC
Availability Zone 1...
Availability Zone 2
Availability Zone 3
Public subnet
NAT Gateway
Public subnet
NAT Gateway
Public subnet
NAT Gateway
Kubeflow user auth redirect
Private subnet
Private subnet
Private subnet
Experimental Kubeflow users
kubectl
Amazon EKS
Kubeflow
Kubeflow
Kubeflow
Kubeflow Dashboard UI - Kubeflow Notebooks - Kubeflow Pipelines - Kubeflow Training Operators - Kubeflow Metadata Store - Katib ...
Dex Auth on K8S - User secrets on K8S secrets - MySQL K8S on EBS - MinIO K8S on EBS
Amazon EBS
Amazon EBS
Amazon EBS
Internet gateway

Figure 2: Kubeflow can be launched far faster on AWS Elastic Kubernetes Service (EKS) than on a local K8s cluster, not least because AWS has put a huge amount of effort into preparing this setup. © AWS

should never see in its plain vanilla form.

Instead, the idea is for distributors to prepare Kubeflow, coordinate the individual components, and then distribute a ready-to-run package (e.g., with the Helm package manager). Some ready-made implementations are on the market, including deployKF (**Figure 3**), but an attempt to get it running on a local K8s cluster quickly shows that even if you do manage to get it to work, you will likely be too exhausted by the process to do anything with the final results.

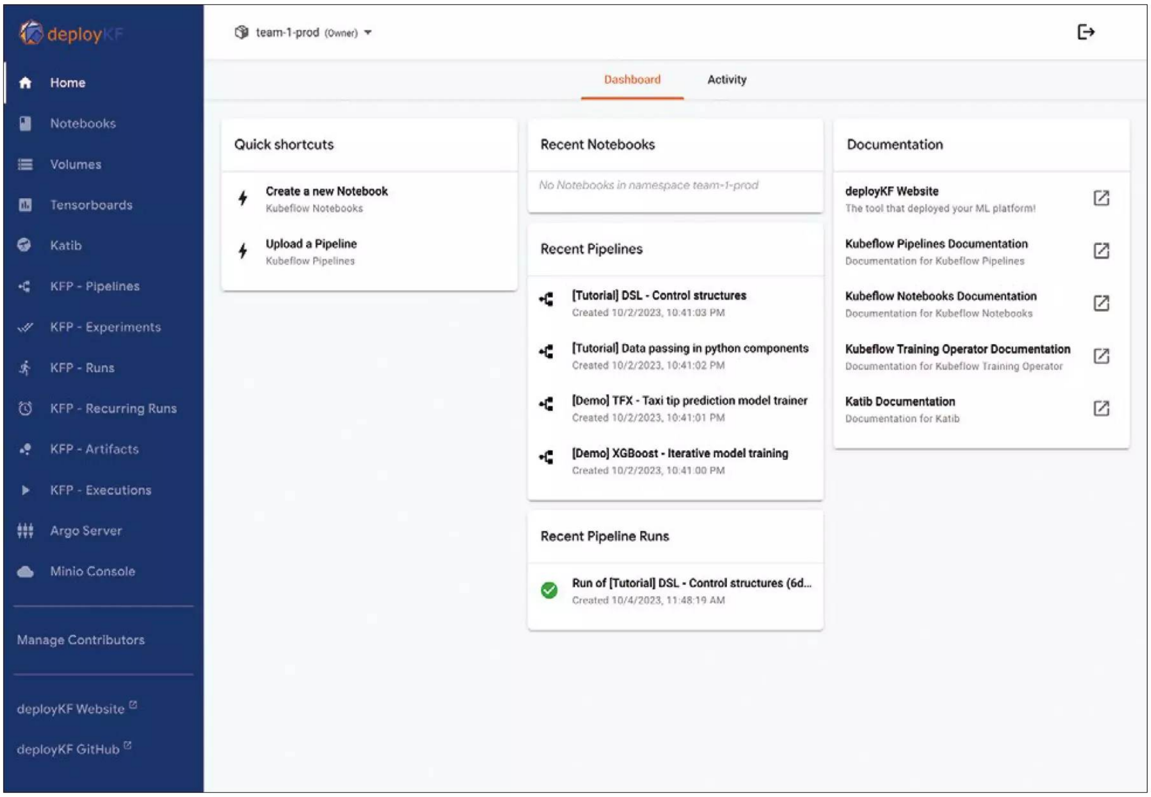
With the enormity of this task in mind, hyperscalers have turned their attention to Kubeflow and are actively supporting it. Azure, AWS, and Google have the required hardware in place and ready-made distributions of Kubernetes in their portfolios that are perfectly tailored to their own platform. Because these organizations have enough manpower to prepare Kubeflow, it should come as little surprise that the Kubeflow source code contains a number of commits from AWS, Azure, and Google that contain tweaks here and there to get Kubeflow up and running quickly on the respective platforms.

If you have no previous experience with Kubeflow, though, you are still likely to fail. Although developers can avoid at least most of the Kubeflow complexity, some additional work from the interaction between Kubeflow and your choice of platform still needs to be done (e.g., user admin). Although Kubeflow offers user administration, it can also be easily replaced by identity and access management (IAM) in AWS thanks to a kind of plugin principle. Kubeflow users can then be managed directly by Amazon Cognito. If you are unfamiliar with the cornucopia of services on the target platform, you will probably lose your way in the maze of options and possibilities.

Moreover, Kubeflow offers several starting points for the connection of external services instead of integrated solutions. For example, Kubeflow uses MinIO as S3-style storage by default. For deployment on AWS, though, native AWS S3 would make more sense. Possibilities upon possibilities. Developers who just want to use Kubeflow will end up not seeing the wood for the trees.

A word of warning about AWS at this point: The final setup on AWS as described in this article will generate costs of around EUR60 per day. The price will vary depending on the region you use for the AWS setup, your currency, and the resource types. For example, if you want to go whole hog here and use expensive GPU instances for your AI workloads, you can easily exceed this price-per-day barrier.

If you want to experiment with Kubeflow, you can do so on AWS, and Kubeflow will also run perfectly in production operations on AWS. However, I would strongly recommended not running any test setups



Quick shortcuts

Create a new Notebook
Kubeflow Notebooks

Upload a Pipeline
Kubeflow Pipelines

Recent Notebooks

No Notebooks in namespace team-1-prod

Recent Pipelines

[Tutorial] DSL - Control structures
Created 10/2/2023, 10:41:03 PM

[Tutorial] Data passing in python components
Created 10/2/2023, 10:41:02 PM

[Demo] TFX - Taxi tip prediction model trainer
Created 10/2/2023, 10:41:01 PM

[Demo] XGBoost - Iterative model training
Created 10/2/2023, 10:41:00 PM

Recent Pipeline Runs

Run of [Tutorial] DSL - Control structures (6d...
Created 10/4/2023, 11:48:19 AM

Documentation

deployKF Website
The tool that deployed your ML platform!

Kubeflow Pipelines Documentation
Documentation for Kubeflow Pipelines

Kubeflow Notebooks Documentation
Documentation for Kubeflow Notebooks

Kubeflow Training Operator Documentation
Documentation for Kubeflow Training Operator

Katib Documentation
Documentation for Katib

Figure 3: Kubeflow comes with several distributions – in this example, deployKF for local clusters. However, if you are looking for a quick-start approach, hyperscalers are a better choice. © deployKF

you do not need, because the AWS dollar meter will just keep ticking along. Moreover, I would strongly recommend defining a budget limit in the AWS billing tool so that you will at least see an email warning if you exceed certain amounts.

Preparations

Before you can even get started with Kubeflow, you need to make a few preparations in AWS [2]. In the following discussion, I assume you have a brand new AWS account with two-factor authentication in place and a stored payment method. Although the free trial allocation at AWS can also be used with Kubeflow, certain services cannot be used without a credit card, and Kubeflow requires some of these services: no credit card, no Kubeflow.

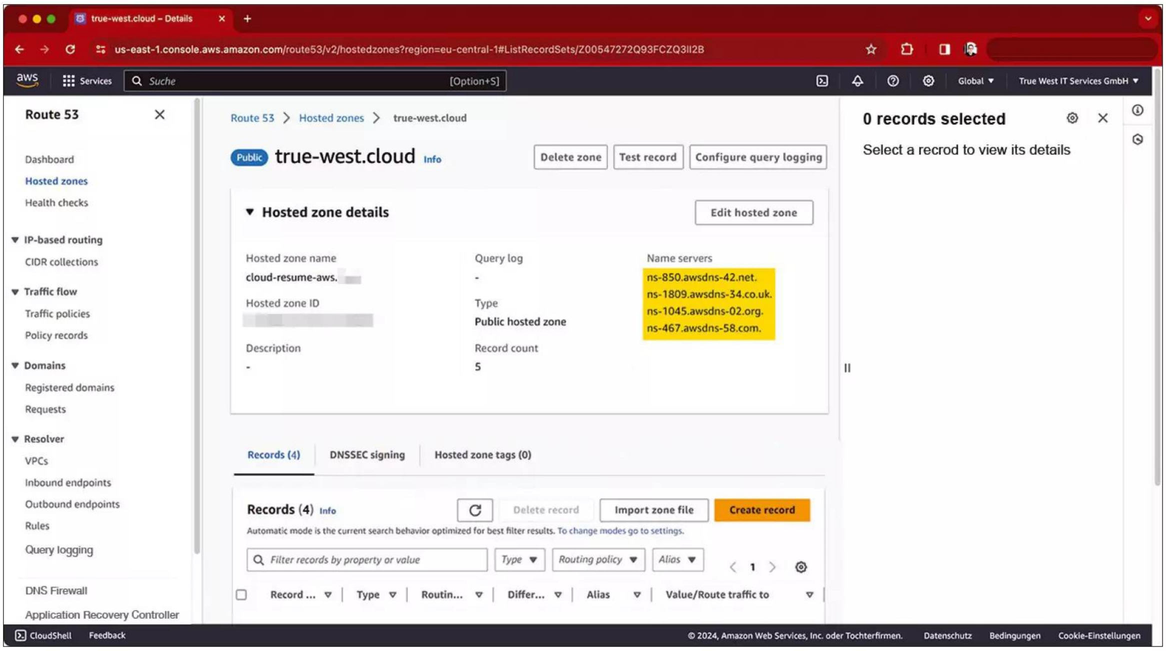
Once you are all warm and cosy in your new AWS account, it's time for the first real task. To access the Kubeflow instance to be built, Kubeflow needs to be accessible from the network. The deployment tools available for Kubeflow deployment on AWS assume that a separate domain is available for this purpose in Amazon Route 53 (i.e., the AWS DNS manager with an integrated load balancer), which is the only way the Kubeflow setup tools will later be able to create a virtual load balancer on AWS; it then also has a public IP address and is accessible by a defined hostname. In theory, a subdomain of an existing domain that is sub-delegated by DNS is all you need at this point.

However, because *.cloud* domains in particular are quite cheap to obtain, my experience is that it makes more sense to dedicate a separate domain to your own Kubeflow adventures and place it completely under the auspices of AWS. Kubeflow then creates a subdomain for this domain in Route 53 in the scope of the installation, which also sets up direct access to the load balancer. Importantly, if you configured your own domain for use in Route 53 or delegated a subdomain of an existing domain, you will need the ID of the domain in Route 53, which you can discover by opening the hosted domain there and clicking *Hosted zones* | *View details*. The required information can be found in the *Hosted zone ID* field (Figure 4).

Starting EKS

Because Kubeflow is based on Kubernetes, a working EKS cluster is a mandatory requirement for Kubeflow on AWS. I recommend the `eksctl` command-line tool, which communicates directly with the AWS EKS API and creates the required cluster in just a few minutes. The following example assumes that all commands are run on Ubuntu 22.04, but most of the commands will probably work on other distributions with just a few minor changes. To use `eksctl` on Ubuntu, first install the required binary directly from Amazon with the commands in Listing 1.

A call to `eksctl --version` should then display the version information of the tool on the console. Although `eksctl` is now basically ready for use, it lacks the access credentials for AWS. You can store these with the `aws configure` command. AWS prompts for the Access Key ID and the Secret Access Key, two pieces of information that can be found directly on the overview page of your AWS account. Depending on your personal preferences, you can also specify a default region and a default output format, but it is not


true-west.cloud – Details
us-east-1.console.aws.amazon.com/route53/v2/hostedzones?region=eu-central-1#ListRecordSets/Z00547272Q93FCZQ3II2B
aws
Services
Suche
[Option+S]
Global
True West IT Services GmbH
Route 53
Dashboard
Hosted zones
Health checks
IP-based routing
CIDR collections
Traffic flow
Traffic policies
Policy records
Domains
Registered domains
Requests
Resolver
VPCs
Inbound endpoints
Outbound endpoints
Rules
Query logging
DNS Firewall
Application Recovery Controller
Route 53 > Hosted zones > true-west.cloud
Public
true-west.cloud
Info
Delete zone
Test record
Configure query logging
Hosted zone details
Edit hosted zone
Records (4)
DNSSEC signing
Hosted zone tags (0)
Delete record
Import zone file
Create record
Filter records by property or value
Type
Routing policy
Alias
1
Record ...
Type
Routin...
Differ...
Alias
Value/Route traffic to
0 records selected
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. oder Tochterfirmen.
Datenschutz
Bedingungen
Cookie-Einstellungen

Hosted zone name
cloud-resume-aws.[redacted]

Hosted zone ID
[redacted]

Description
-

Query log
-

Type
Public hosted zone

Record count
5

Name servers
ns-850.awsdns-42.net.
ns-1809.awsdns-34.co.uk.
ns-1045.awsdns-02.org.
ns-467.awsdns-58.com.

Select a recrod to view its details

Records (4) Info

Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.

Figure 4: For the prepared Cognito Kubeflow setup to work on EKS, you need the ID of the zone in Route 53; the entry for the Kubeflow load balancer will later reside in a dedicated subzone here.

mandatory. Once AWS and `eksctl` are ready to go, the next step is to create an EKS cluster (**Listing 2**).

Some of the details in the command shown here are variable; you can or need to adjust them – in particular, the value for `AWS_ACCOUNT`, which determines which of the stored sets of access credentials `eksctl` uses. The parameter for `--node-type` can also be changed. The advice, though, is only to choose a smaller instance size if this is a test cluster. In contrast, production systems will possibly need an even larger instance type depending on the workload, although keeping a watchful eye on the invoice amounts in AWS is definitely recommended.

Once you have completed this step, the next step is to create the Kubeflow cluster by first preparing the folder with the scripts and integration components for AWS from the Kubeflow Git directory and navigating to a subfolder of the freshly checked-out source code:

```
$ git clone https://github.com/awslabs/➥
                         kubeflow-manifests/
$ cd kubeflow-manifests/tests/e2e/➥
    utils/cognito_bootstrap/
[... edit config.yaml ...]
```

Of particular interest is the file `config.yaml`, although it is largely empty by default. You need to modify several values in this file: `hostedZoneId` and `name` below the `route53.rootDomain` element and `cluster.name`, `cluster.region`, and `name` below the `cognitoUserpool` element. The last value is variable, but a mnemonic name is recommended. Note that the names that follow `cluster.name` and `cluster.region` must match the values you used when creating the EKS cluster.

Finally, change the value of `name` below the `route53.subDomain` element to define the subdomain in which the Kubeflow API will later be accessible, which is why it must be a subdomain of the main domain stored in Route 53 (e.g., `true-west.cloud` and `kubeflow.true-west.cloud` in this example).

Finally, move up two directory levels and launch the integration script:

```
$ cd kubeflow-manifests/tests/e2e/
$ PYTHONPATH=.. python utils/➥
  cognito_bootstrap/➥
  cognito_pre_deployment.py
[... Pre-deployment run ...]
```

The script automatically creates resources in AWS Route 53 and AWS Cognito that Kubeflow will later need to create a functional load balancer and integrate with Cognito. The script also automatically creates the subdomain you need in Route 53 and extends the `config.yaml` file (which you edited manually previously) so that it contains valid values at this point in time. After the pre-deployment step, you will later have – as the experienced administrator will already have guessed – a post-deployment step.

Please note that depending on the time of day and the instances used in AWS, both creating the EKS cluster and handling the required preparations can take 20 minutes or longer. However, unless the tools explicitly display an error message, you do not need to worry; AWS is just a little slow at times. Once this preparatory step has been completed, continue with the Kubeflow deployment.

Creating Namespaces

A minor adjustment still needs to be made because of the way in which Kubeflow implements a form of multiclient capability internally and how the integration component is implemented in Cognito.

Under the hood, Kubeflow is based on the idea of namespaces. The term is probably familiar to people from the Kubernetes universe, because namespaces also exist there. A Kubernetes namespace is always part of a Kubeflow namespace, as well, but the Kubeflow namespace also includes a user configuration and a set of rules that grant the newly created Cognito user access to a Kubeflow namespace.

Kubeflow creates a namespace for a user by default if the user does not have one when they first log in. This function is disabled in the default Kubeflow configuration, though. After

Listing 1: Installing eksctl and AWS

```
$ ARCH=amd64
$ PLATFORM=$(uname -s)_$ARCH
$ curl -sLO "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl_$PLATFORM.tar.gz"
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
$ tar -xzf "eksctl_$PLATFORM.tar.gz" -C /tmp && rm "eksctl_$PLATFORM.tar.gz"
$ unzip -u awscliv2.zip
$ sudo mv /tmp/eksctl /usr/local/bin
$ sudo ./aws/install
```

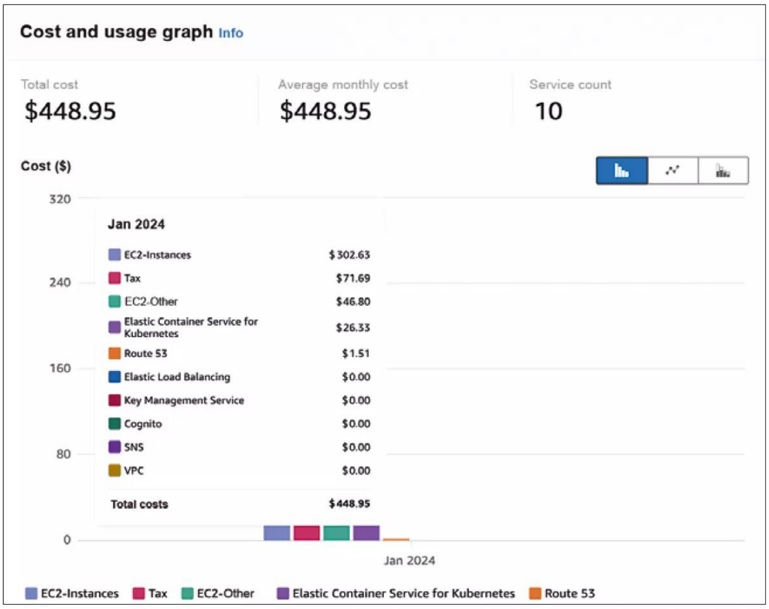
Listing 2: Creating an EKS Cluster

```
export AWS_ACCOUNT=<Access Key ID>
export CLUSTER_REGION=eu-central-1
export CLUSTER_NAME=kubeflow-1
export PROFILE_NAME=kubeflow-user
export PROFILE_CONTROLLER_POLICY_NAME=kubeflow-user

eksctl create cluster --name ${CLUSTER_NAME} --version 1.25 --region ${CLUSTER_REGION} --nodegroup-name
  linux-nodes --node-type <m5.xlarge> --nodes 5 --nodes-min 5 --nodes-max 10 --managed --with-oidc

eksctl create iamserviceaccount --region ${CLUSTER_REGION} --name ebs-csi-controller-sa --namespace
  kube-system --cluster ${CLUSTER_NAME} --attach-policy-arn arn:aws:iam::aws:policy/service-role/
  AmazonEBSCSIDriverPolicy --approve --role-only --role-name AmazonEKS_EBS_CSI_DriverRole

eksctl create addon --name aws-ebs-csi-driver --cluster ${CLUSTER_NAME} --service-account-role-arn
  arn:aws:iam::$(aws sts get-caller-identity --query Account --output text):role/AmazonEKS_EBS_CSI_
  DriverRole --force
```



Cost and usage graph Info

Total cost $448.95

Average monthly cost $448.95

Service count 10

Cost ($)

Jan 2024	
EC2-Instances	$ 302.63
Tax	$71.69
EC2-Other	$46.80
Elastic Container Service for Kubernetes	$26.33
Route 53	$1.51
Elastic Load Balancing	$0.00
Key Management Service	$0.00
Cognito	$0.00
SNS	$0.00
VPC	$0.00
Total costs	$448.95

Figure 5: **Caution is advised when experimenting with Kubeflow on EKS: If you forget a test setting, you will very quickly generate horrendous bills.**

a setup without any customization, users could therefore log into Kubeflow but not work with it. To change this behavior, change to the subdirectory in the `kubeflow-manifests/` folder,

```
$ cd kubeflow-manifests/charts/apps/↩
  central-dashboard/templates/ConfigMap/
```

and change the value of `CD_REGISTRATION_FLOW` to true in the `centraldashboard-parameters-kubeflow-ConfigMap.yaml` file. This adjustment is followed by the Kubeflow deployment:

```
$ cd kubeflow-manifests/
$ make deploy-kubeflow ↩
  INSTALLATION_OPTION=kustomize ↩
  DEPLOYMENT_OPTION=cognito
```

Again, you will need to be patient: More than 70 different services and well over 100 containers need to be downloaded and started in a process that usually takes 15 minutes or more. The wait is worthwhile: Kubeflow will basically be rolled out and ready for use. Of course, you can't access it as yet because the Cognito configuration and the associated load balancer are missing.

The last step of the process is to create both of these elements to conclude the setup:

```
$ cd kubeflow-manifests/tests/e2e/
$ PYTHONPATH=.. python utils/↩
  cognito_bootstrap/↩
  cognito_post_deployment.py
```

The Kubeflow dashboard is now available for login on *kubeflow.<subdomain>* (e.g., *https://kubeflow.k8s.true-west.cloud* here). Now all that is missing is the user account in AWS Cognito, and you can create this in the normal way from the AWS GUI.

Unlike a DIY Kubeflow (e.g., on the basis of deployKF), all the components will work smoothly immediately after completing the setup in the EKS-based variant. Pipelines can be created and managed, just like new notebooks or server applications to serve (trained) models for sharing with the outside world. Kubeflow even comes with some example pipelines out of the box.

It is beyond the scope of this article to go into detail about how Kubeflow works and how models can be created and trained in it, but some sample projects and comprehensive instructions can be found online.

Cleanup

Cleaning up after working with Kubeflow does not involve anything like the number of hoops you had to jump through to install the environment. All it takes is:

```
$ kubectl get svc --all-namespaces
$ kubectl delete svc <service>
$ eksctl delete cluster ↩
  --name ${CLUSTER_NAME}
```

The first command displays all the services stored in the cluster. `kubectl` works smoothly after creating a cluster with `eksctl`, because `eksctl` stores the access data locally for each newly created cluster and sets up environment variables to ensure that the right credentials are used.

The last two lines delete services you no longer need and clear the cluster again when done, provided it was set up as described in the example. If the `delete` command returns the value *0*, the cluster has been removed successfully and is no longer using any resources that would add to your bill (Figure 5).

According to the process described, a Kubeflow installation can be set up from scratch at any time. The important thing is to reset `config.yaml` for Cognito before the pre-deployment step so that it only contains the information you need for the domain you will be using. The deployment tools modify the file several times during setup. After completing the Kubeflow deployment, it can no longer be used to set up another new cluster. ■

Info

[1] Kubeflow: [https://www.kubeflow.org]
[2] Kubeflow-AWS Cognito guide: [https://awslabs.github.io/kubeflow-manifests/docs/deployment/cognito/]

The Author

Martin Loschwitz is the founder and managing director of True West IT Services GmbH, which offers scalable IT infrastructure based on OpenStack and Kubernetes.

Traffic analysis with mitmproxy

Traffic Monitor

The mitmproxy tool puts interactive traffic analysis in your hands.

By Holger Reibold

Companies that provide web services for the outside world in their own infrastructure are exposed to a variety of threats. The developers of the open source mitmproxy tool describe it as the Swiss army knife for debugging, testing, data protection analysis, and penetration testing HTTP(S) connections. I show you how mitmproxy can be a useful addition to your security toolbox.

Man in the Middle

When most people hear the term “proxy,” they probably think of legacy proxy servers that act as gateways connecting local networks to the global network or as go-betweens protecting local clients against external access (e.g., NGINX, Squid, and WinGate come to mind). Given the name, mitmproxy [1] could be assumed to be in the same category. However, the tool takes a different approach by specializing in HTTP(S) traffic analysis. Like Wireshark, the software is more of a sniffer that records the data traffic between the HTTP client and server and enables analysis by doing so.

The *mitm* part of the name hints at its functionality: mitmproxy acts as a man-in-the-middle (MITM) proxy that intercepts and modifies HTTP and HTTPS data traffic. You can record the HTTP conversation for later analysis, although the tool is limited to the protocol-specific data exchange. Unlike Wireshark and other sniffers, no other data is logged.

Mitmproxy can also act as a reverse proxy and forward data traffic to a specific server. Script-based manipulation of HTTP traffic is also an option, for which you can use simple Python scripts. Interaction with third-party applications for automatic manipulation or visualization is also possible with the Python API. Mitmproxy can generate SSL/TLS certificates for interception, as well.

Basic Principles

A basic understanding of how mitmproxy works is useful if you want to work effectively in the environment. For example, take a look at HTTPS-protected access by a client to a web server. The client uses the HTTP command:

```
CONNECT server.en:443 HTTP/1.1
```

A legacy proxy server cannot manipulate SSL-/TLS-encrypted data traffic but simply forwards the request to the target system; it thus lives up to its name as an authorized agent. When you use mitmproxy, the HTTPS proxy sits between the client and server with the classic man-in-the-middle approach. For the client, mitmproxy looks like a server, while pretending to be the client for the server, which allows mitmproxy to decrypt the data traffic from both sides.

Lead Image © r lakhesis, 123RF.com

The challenge for mitmproxy is that the certification authority's system is designed to prevent precisely this type of attack by allowing a trusted third party to sign a server's certificates cryptographically to verify its legitimacy. If any discrepancies are noted, the connection is interrupted, which is why it is often difficult to analyze secure connections.

The mitmproxy developers used a trick to solve this problem. The software itself acts as a trustworthy certificate authority (CA). To do so, mitmproxy comes with a complete CA implementation that generates all the required certificates on the fly. For the client to trust these certificates, you need to register mitmproxy manually as a trusted CA.

Mitmproxy then needs to overcome further challenges to inject the environment between the client and server without being noticed. For example, the domain name of the remote party must be determined so that it can be used in the intercept certificate. To do so, mitmproxy uses upstream certificate sniffing. The tool also extracts the common name (CN) from the upstream certificate and the subject alternative names (SANs). It also cleverly works around the server name indication (SNI) handover.

The following processes take place for HTTPS connections with an intermediate mitmproxy (Figure 1):

1. The client establishes a connection to mitmproxy and generates an HTTP `CONNECT` request.
2. Mitmproxy responds with a *200 Connection Established* message, simulating a `CONNECT` pipe being opened.
3. The client opens an SSL/TLS connection on the assumption that it is talking to the desired server. On doing so, it states the SNI for the hostname.
4. Mitmproxy opens the connection to the server and sets up a secure connection with the SNI hostname specified by the client.
5. The target system responds with a certificate containing the CN and SAN values required to create the interception certificate.
6. Mitmproxy generates the interception certificate and continues the client SSL/TLS handshake interrupted in step 3.
7. The client sends the request through the open connection.
8. Mitmproxy forwards the request to the server through the connection initiated in step 4.

This simplified process sequence shows the sophistication of the actions performed by mitmproxy just to slip into position between the client and the server.

Getting Started

Mitmproxy is available for Linux, macOS, and Windows. Standalone binaries are available for Windows and Linux, with distribution-specific packages for various Linux distributions (e.g., Arch, Debian, Ubuntu, and Kali). After installing on Windows, `mitmproxy`, `mitmdump`, and `mitmweb` are added to `PATH` and can be called from the command line. In principle, the Linux packages will also run on Windows Subsystem for Linux (WSL).

If you prefer to use a Docker container [2], use the following command to launch the mitmproxy terminal interface:

```
docker run --rm -it ↲
  -v ~/.mitmproxy:/home/mitmproxy/↲
       .mitmproxy ↲
  -p 8080:8080 mitmproxy/mitmproxy
```

To make sure the client's web traffic is routed through mitmproxy, you need

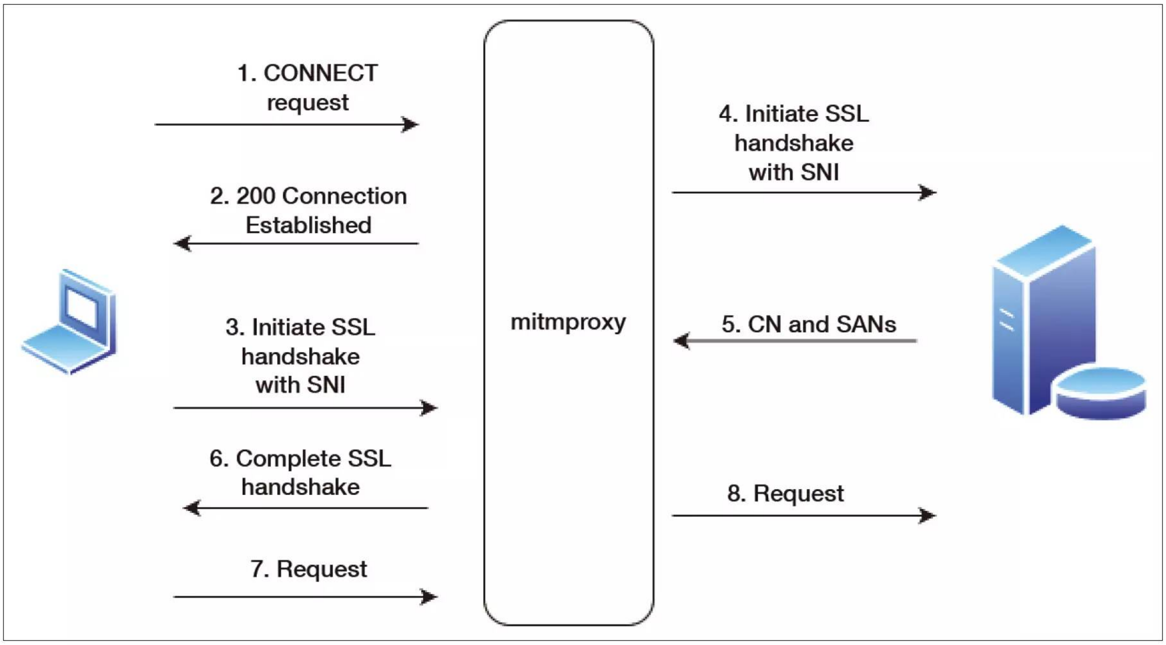

1. CONNECT request
2. 200 Connection Established
3. Initiate SSL handshake with SNI
6. Complete SSL handshake
7. Request
mitmproxy
4. Initiate SSL handshake with SNI
5. CN and SANs
8. Request

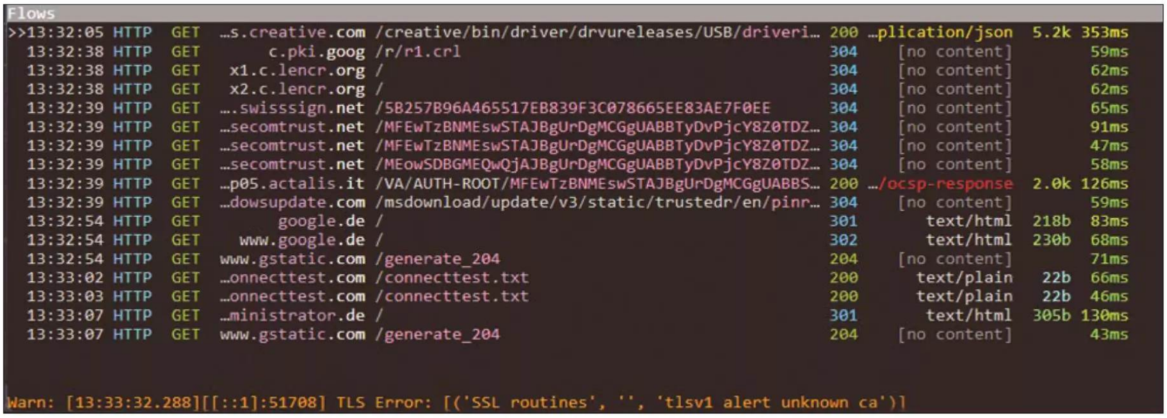
Figure 1: **Mitmproxy's man-in-the-middle position allows web traffic to be intercepted and manipulated.**

to adjust the global settings when installing the software locally and enter a proxy address of *127.0.0.1:8080*. You could also use a proxy auto-configuration (PAC) file to simplify the global network configuration; the required files are available online [3].

If you try to access the Internet with your web browser after configuring the proxy, the attempt will fail. Your browser displays a *Your connection is not private* error message. The reason is simple: The browser does not consider the intermediary proxy to be trustworthy because it considers the mitmproxy certificate invalid. Once the proxy server is activated, you can pick up the required certificate from the *mitm.it* website. Valid certificates for all supported platforms are available there. On Windows, use the import wizard to deploy the certificate. On Linux, the easiest way to import the certificate is with the command:

```
sudo security add-trusted-cert -d ➥
  -p ssl -p basic ➥
  -k /Library/Keychains/System.keychain ➥
  mitmproxy-ca-cert.pem
```

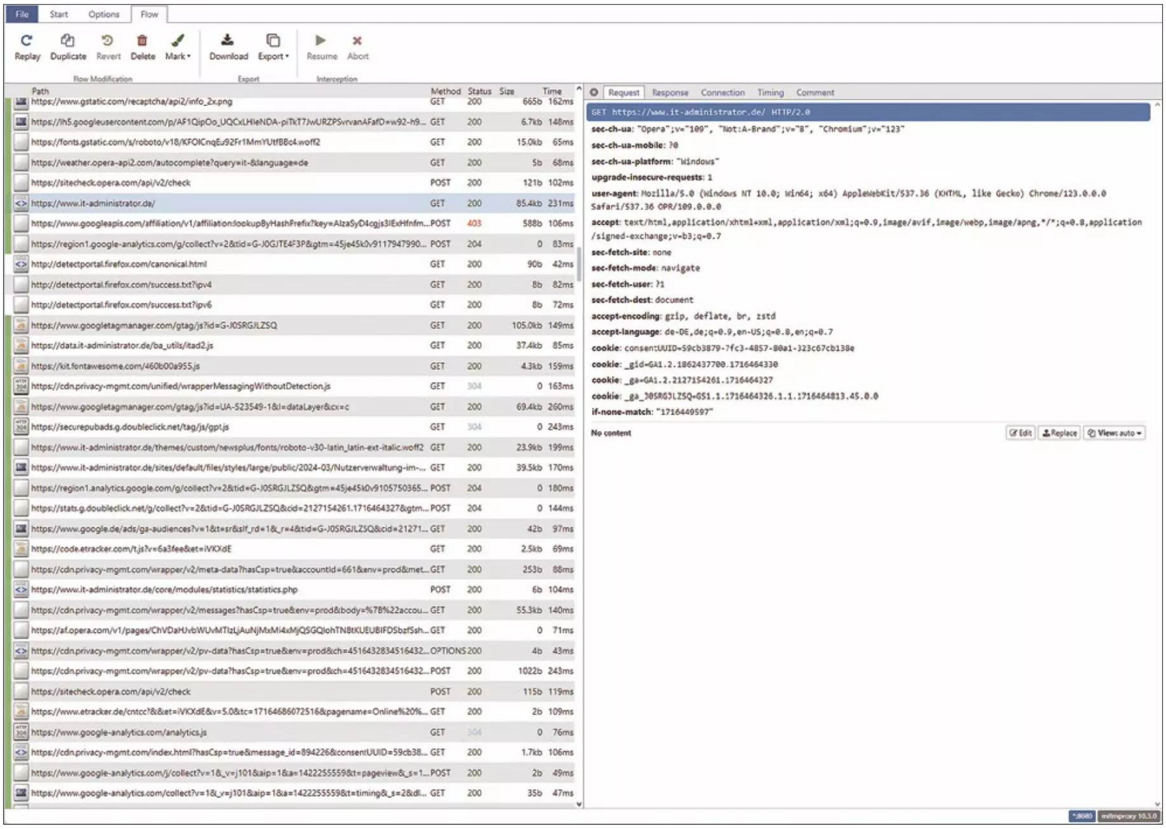
Installation instructions are available online [4] for all other platforms. You can then view and, if needed, edit



```
Flows
>>13:32:05 HTTP  GET  …s.creative.com /creative/bin/driver/drvureleases/USB/driveri… 200 …plication/json  5.2k 353ms
  13:32:38 HTTP  GET       c.pki.goog /r/r1.crl                                      304   [no content]        59ms
  13:32:38 HTTP  GET   x1.c.lencr.org /                                              304   [no content]        62ms
  13:32:38 HTTP  GET   x2.c.lencr.org /                                              304   [no content]        62ms
  13:32:39 HTTP  GET  ….swisssign.net /5B257B96A465517EB839F3C078665EE83AE7F0EE       304   [no content]        65ms
  13:32:39 HTTP  GET  …secomtrust.net /MFEwTzBNMEswSTAJBgUrDgMCGgUABBTyDvPjcY8Z0TDZ… 304   [no content]        91ms
  13:32:39 HTTP  GET  …secomtrust.net /MFEwTzBNMEswSTAJBgUrDgMCGgUABBTyDvPjcY8Z0TDZ… 304   [no content]        47ms
  13:32:39 HTTP  GET  …secomtrust.net /MEowSDBGMEQwQjAJBgUrDgMCGgUABBTyDvPjcY8Z0TDZ… 304   [no content]        58ms
  13:32:39 HTTP  GET  …p05.actalis.it /VA/AUTH-ROOT/MFEwTzBNMEswSTAJBgUrDgMCGgUABBS… 200 …/ocsp-response  2.0k 126ms
  13:32:39 HTTP  GET  …dowsupdate.com /msdownload/update/v3/static/trustedr/en/pinr… 304   [no content]        59ms
  13:32:54 HTTP  GET        google.de /                                              301      text/html  218b  83ms
  13:32:54 HTTP  GET    www.google.de /                                              302      text/html  230b  68ms
  13:32:54 HTTP  GET  www.gstatic.com /generate_204                                  204   [no content]        71ms
  13:33:02 HTTP  GET  …onnecttest.com /connecttest.txt                               200     text/plain   22b  66ms
  13:33:03 HTTP  GET  …onnecttest.com /connecttest.txt                               200     text/plain   22b  46ms
  13:33:07 HTTP  GET  …ministrator.de /                                              301      text/html  305b 130ms
  13:33:07 HTTP  GET  www.gstatic.com /generate_204                                  204   [no content]        43ms


Warn: [13:33:32.288][[::1]:51708] TLS Error: [('SSL routines', '', 'tlsv1 alert unknown ca')]
```

Figure 2: You can follow the client's communication on the console and interactively intervene in the data exchange.



Path	Method	Status	Size	Time
https://www.gstatic.com/recaptcha/api2/info_2x.png	GET	200	665b	162ms
https://lh5.googleusercontent.com/p/AF1QipOo_UQCxLHIeNDA-piTkT7JwURZPSvrvanAFafD=w92-h9...	GET	200	6.7kb	148ms
https://fonts.gstatic.com/s/roboto/v18/KFOlCnqEu92Fr1MmYUtfBBc4.woff2	GET	200	15.0kb	65ms
https://weather.opera-api2.com/autocomplete?query=it-&language=de	GET	200	5b	68ms
https://sitecheck.opera.com/api/v2/check	POST	200	121b	102ms
https://www.it-administrator.de/	GET	200	85.4kb	231ms
https://www.googleapis.com/affiliation/v1/affiliation:lookupByHashPrefix?key=AIzaSyD4cgjs3IExHfnfm...	POST	403	588b	106ms
https://region1.google-analytics.com/g/collect?v=2&tid=G-J0GJTE4F3P&gtm=45je45k0v9117947990...	POST	204	0	83ms
http://detectportal.firefox.com/canonical.html	GET	200	90b	42ms
http://detectportal.firefox.com/success.txt?ipv4	GET	200	8b	82ms
http://detectportal.firefox.com/success.txt?ipv6	GET	200	8b	72ms
https://www.googletagmanager.com/gtag/js?id=G-J0SRGJLZSQ	GET	200	105.0kb	149ms
https://data.it-administrator.de/ba_utils/itad2.js	GET	200	37.4kb	85ms
https://kit.fontawesome.com/460b00a955.js	GET	200	4.3kb	159ms
https://cdn.privacy-mgmt.com/unified/wrapperMessagingWithoutDetection.js	GET	304	0	163ms
https://www.googletagmanager.com/gtag/js?id=UA-523549-1&l=dataLayer&cx=c	GET	200	69.4kb	260ms
https://securepubads.g.doubleclick.net/tag/js/gpt.js	GET	304	0	243ms
https://www.it-administrator.de/themes/custom/newsplus/fonts/roboto-v30-latin_latin-ext-italic.woff2	GET	200	23.9kb	199ms
https://www.it-administrator.de/sites/default/files/styles/large/public/2024-03/Nutzerverwaltung-im-...	GET	200	39.5kb	170ms
https://region1.analytics.google.com/g/collect?v=2&tid=G-J0SRGJLZSQ&gtm=45je45k0v9105750365...	POST	204	0	180ms
https://stats.g.doubleclick.net/g/collect?v=2&tid=G-J0SRGJLZSQ&cid=2127154261.1716464327&gtm...	POST	204	0	144ms
https://www.google.de/ads/ga-audiences?v=1&t=sr&slf_rd=1&_r=4&tid=G-J0SRGJLZSQ&cid=21271...	GET	200	42b	97ms
https://code.etracker.com/t.js?v=6a3fee&et=iVKXdE	GET	200	2.5kb	69ms
https://cdn.privacy-mgmt.com/wrapper/v2/meta-data?hasCsp=true&accountId=661&env=prod&met...	GET	200	253b	88ms
https://www.it-administrator.de/core/modules/statistics/statistics.php	POST	200	6b	104ms
https://cdn.privacy-mgmt.com/wrapper/v2/messages?hasCsp=true&env=prod&body=%7B%22accou...	GET	200	55.3kb	140ms
https://af.opera.com/v1/pages/ChVDaHJvbWUvMTIzLjAuNjMxMi4xMjQSGQIohTN8tKUEUBIFDSbzfSsh...	GET	200	0	71ms
https://cdn.privacy-mgmt.com/wrapper/v2/pv-data?hasCsp=true&env=prod&ch=4516432834516432...	OPTIONS	200	4b	43ms
https://cdn.privacy-mgmt.com/wrapper/v2/pv-data?hasCsp=true&env=prod&ch=4516432834516432...	POST	200	1022b	243ms
https://sitecheck.opera.com/api/v2/check	POST	200	115b	119ms
https://www.etracker.de/cntcc?&&et=iVKXdE&v=5.0&tc=17164686072516&pagename=Online%20%...	GET	200	2b	109ms
https://www.google-analytics.com/analytics.js	GET	304	0	76ms
https://cdn.privacy-mgmt.com/index.html?hasCsp=true&message_id=894226&consentUUID=59cb38...	GET	200	1.7kb	106ms
https://www.google-analytics.com/j/collect?v=1&_v=j101&aip=1&a=1422255559&t=pageview&_s=1...	POST	200	2b	49ms
https://www.google-analytics.com/collect?v=1&_v=j101&aip=1&a=1422255559&t=timing&_s=2&dl...	GET	200	35b	47ms

```
GET https://www.it-administrator.de/ HTTP/2.0
sec-ch-ua: "Opera";v="109", "Not:A-Brand";v="8", "Chromium";v="123"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
upgrade-insecure-requests: 1
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0
Safari/537.36 OPR/109.0.0.0
accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application
/signed-exchange;v=b3;q=0.7
sec-fetch-site: none
sec-fetch-mode: navigate
sec-fetch-user: ?1
sec-fetch-dest: document
accept-encoding: gzip, deflate, br, zstd
accept-language: de-DE,de;q=0.9,en-US;q=0.8,en;q=0.7
cookie: consentUUID=59cb3879-7fc3-4857-80a1-323c67cb138e
cookie: _gid=GA1.2.1862437700.1716464330
cookie: _ga=GA1.2.2127154261.1716464327
cookie: _ga_J0SRGJLZSQ=GS1.1.1716464326.1.1.1716464813.45.0.0
if-none-match: "1716449597"

No content
```

Figure 3: The web interface makes analyzing and manipulating web traffic easy.

the details in the certificate manager belonging to your choice of operating system.

Web GUI

You can now track the exact HTTP commands in the mitmproxy console (Figure 2), but analyzing these commands is not particularly user friendly. The mitmproxy web interface can help, which is accessed at *http://127.0.0.1:8081* by default. In the web GUI, mitmproxy shows the HTTP command exchange history. To install the required certificate on the client system, go to *File* | *Install Certificates*.

The *Options* tab offers various customization options for the web interface that mainly relate to the display, but you can also work with block lists and add an empty response for specific requests. Mitmproxy supports the following modes in addition to the *Regular* default mode:

- *Transparent*
- *WireGuard*
- *Reverse Proxy*
- *Upstream Proxy*
- *SOCKS Proxy*
- *DNS Server*

The choice of mode depends on whether you want to monitor the traffic from a client or a single server. If you want to examine the traffic reaching your own web server from the Internet, you need to enable the *Reverse Proxy* variant. In this case, mitmproxy acts like a "normal" server that fields the requests from the Internet or from an analysis client and passes them on to the target system.

To make it easier to find the goodies, switch to the *Start* tab and enter a search term, which is a good way of restricting the view to specific sources, services, or file types. The good thing about this is that if the web GUI is open and you access the server in a second browser window, you can trace the commands in mitmproxy in real time.

The individual requests and responses can be examined both at the console level and in the web interface, but the web interface is far more convenient. You can play back the logged exchanges or select *Edit* to edit requests and forward them to the target server after doing so. It is easy to apply filters at the console level, intercept the traffic, and save the markup locally.

In view of the huge volumes of data that mitmproxy delivers, it is not always easy to keep track of the content in which you are interested, which is where the *Highlight* function comes in handy. Clicking on the *Highlight* button opens a selection menu where you can highlight specific content.

You can limit the selection to methods or content types or even use a regular expression.

IT Highlights at a Glance

Too busy to wade through press releases and chatty tech news sites? Let us deliver the most relevant news, technical articles, and tool tips – straight to your Inbox.

Linux Update • ADMIN Update • ADMIN HPC

Keep your finger on the pulse of the IT industry.

ADMIN and HPC: bit.ly/HPC-ADMIN-Update
Linux Update: bit.ly/Linux-Update

Photo by Andrew Neel on Unsplash

Advanced Analysis Techniques

One of mitmproxy's special features is the *Interception* function, which lets you intercept and manipulate requests. Intercepting is not normally desirable because it impairs the browsing experience. To keep you from taking down the entire data exchange, mitmproxy uses a different approach and intercepts requests selectively. You can set this up in the web GUI or at the console with the `~u <regex>` flow filter, and you can use the `~q` option to avoid intercepting the responses. An ampersand (&) lets you combine multiple filter options.

Once you have interrupted a request, the next step is to modify it to suit your needs. If you are working with the web GUI, first enable Edit mode and select the entry with the request that you want to edit (e.g., you could manipulate the `Path` or `user-agent` option).

If you want to make the changes at the console level, press *E* to enable Edit mode; in the web GUI, open the *Request* tab (**Figure 3**) and navigate to the entry in which you are interested. Make your changes and exit Edit mode. To resume the interrupted data flow, press the *A* button at the console or click *Resume* in the GUI.

Another special feature of mitmproxy is that you can replay previous data flows repeatedly. The tool supports two types of replay requests:

- Client-side replay retransmits previous client requests to the server.
- Server-side replay replays the server's responses to previously recorded requests.

You can use both variants at the console and in the web GUI. To begin, select the desired requests with filter expressions. To repeat these, press the *r* key at the console or click *Replay* in the GUI. You can modify the request here, too.

Thanks to its modular architecture, you can extend mitmproxy's functionality and draw on the development work of an active community. **Table 1** summarizes the most interesting extensions. For example, the `log-events.py` script runs at console level and generates warnings.

Table 1: Popular mitmproxy Extensions

Script	Description
`log-events.py`	Write messages to the mitmproxy event log.
`http-add-header.py`	Add an HTTP header to each response.
`internet-in-mirror.py`	Mirror all websites.
`commands-simple.py`	Add a user-defined command to the command line.
`shutdown.py`	Shut down and terminate a mitmproxy instance.
`duplicate-modifyreplay.py`	Accept incoming HTTP requests and replay them with modified parameters.
`tcp-simple.py`	Process individual messages from a TCP connection.
`websocket-injectmessage.py`	Inject a WebSocket message into an open connection.

Excluding Domains

When analyzing traffic, it might make sense to ignore certain domains – for two main reasons. First, some of the traffic might be protected by certificate pinning, and second, some of the traffic might simply be beyond your scope of interest.

In the first case, for example, Windows updates or access to Apple's App Store will not work if you have set up mitmproxy in the middle. If some of the content is not relevant to you, simply hide it with the `view_filter` option.

To exclude hosts from monitoring, use the `ignore_hosts` option and specify this with a regular expression in the `host:port string` format. The command line alias is `--ignore-hosts regex`, and the complete command is:

```
mitmproxy ↲
  -ignore-hosts '^example\.de:443$'
```

If you want to restrict recording to specific domains, use the `--allow-hosts` option.

Another great feature of mitmproxy is that the environment has a modular architecture, and you extend the feature set with the help of Python scripts. A major share of the existing feature set is already based on integrated add-ons. The structure and design of these scripts are kept so simple that it is easy to adapt them or develop your own scripts. Almost 30 scripts are currently available for download from the GitHub project site **[5]**.

Conclusions

Mitmproxy is an exciting tool for recording, analyzing, and manipulating HTTP traffic. Thanks to its specific focus, the HTTP proxy is great for checking web requests and responses, which makes it a valuable tool, not only for administrators, but also for developers and penetration testers. ■

Info

[1] mitmproxy: [https://mitmproxy.org]

[2] Docker Hub: [https://hub.docker.com/u/mitmproxy]

[3] PAC files: [https://github.com/kaqu/mitm_mock/tree/master]

[4] Installation instructions: [https://docs.mitmproxy.org/stable/]

[5] Add-ons: [https://github.com/mitmproxy/mitmproxy/tree/main/examples/addons]

DrupalCon
VIENNA2025
14-17 OCTOBER

Mark your Calendar for DrupalCon Vienna 2025!

Get ready to connect, learn, and innovate! From **14 - 17 October 2025, Vienna** hosts **DrupalCon Europe** once again. Whether you're a developer, designer, marketer, or business leader, join us to share ideas, collaborate, and shape the future of Drupal.

Set in the heart of one of Europe's most inspiring cities, DrupalCon Vienna promises a unique blend of history, creativity, and technology. Don't miss out on the opportunity—mark your calendars and **join us for an unforgettable experience!**

Make sure to check the official website and follow us on social media regularly to stay updated and never miss important news!

@DrupalConEur

Photo by Geeky Shots on Unsplash

Run rootless Podman containers as systemd services

Power Up

Running rootless containers is easy with Podman. With Podman Quadlet files, these containers are seamlessly integrated into systemd services. By Koen Vervloesem

Podman since version 4.4 has a native mechanism to start containers automatically by systemd unit files: Podman Quadlet. In this article, I guide you through preparing your system and configuring it to use Quadlets (as the container unit files are called) for your rootless Podman containers. As an example, I'll show you how to set up a reverse proxy in one container and some web services behind the proxy in other containers.

Docker and Podman

If you want to configure which Docker containers should start automatically on your Linux server, Docker Compose [1] is a widely recognized solution. However, within enterprise Linux distributions, Podman [2] has been the preferred container engine for a while. Both Red Hat Enterprise Linux and SUSE Linux Enterprise have adopted Podman, as have their associated community distributions, Fedora and openSUSE.

In contrast to Docker, Podman operates without a continuously running daemon. As a result, Podman is somewhat more lightweight and allows containers to start up faster. Moreover, Podman defaults to running rootless containers, with ordinary user instead of root privileges. SELinux integration is also standard in distributions that support it. Overall, a server running Podman is easier to secure by default than one using Docker.

Podman's compatibility with the Docker API makes it relatively straightforward to use in combination with tools developed for Docker. If you're accustomed to Docker commands, you'll find you can generally substitute the `docker` command with `podman`. For example, `podman ps` lists all running containers, `podman images` shows all downloaded container images, and `podman pull nginx:alpine` downloads the `nginx:alpine` image from one of the registered container repositories.

A Podman Compose [3] solution also processes `docker-compose.yml` files to start your containers with Podman. Another popular solution is configuring systemd services with the appropriate `podman start` commands to manage them. However, the Podman Quadlet [4] alternative simplifies the creation and management of systemd unit files for containers.

Preparing the Container Host

You can choose from various Linux distributions for hosting Podman containers, because many include Podman in their standard repositories. In immutable Linux distributions like Fedora CoreOS [5] and openSUSE MicroOS [6] (Figure 1), Podman comes pre-installed. Both Linux distributions offer a minimal operating system (OS) that automatically updates and provides a stable foundation for running containers.

In this article, I operate Podman on an openSUSE MicroOS server installed with the Base System + Container Runtime Environment package set. If you've only installed the base system, you can still install Podman with:

```
# transactional-update pkg install podman
# reboot
```

A reboot is necessary because, as an immutable Linux distribution, openSUSE MicroOS doesn't allow live changes. On a traditional Linux distribution this reboot isn't required.

Additionally, you should create a user to run all containers:

```
$ sudo useradd -m user
```

The `-m` option instructs `useradd` to create a home directory for the user, where all container data will be stored. To set a password for the newly created user, enter,

```
$ sudo passwd user
```

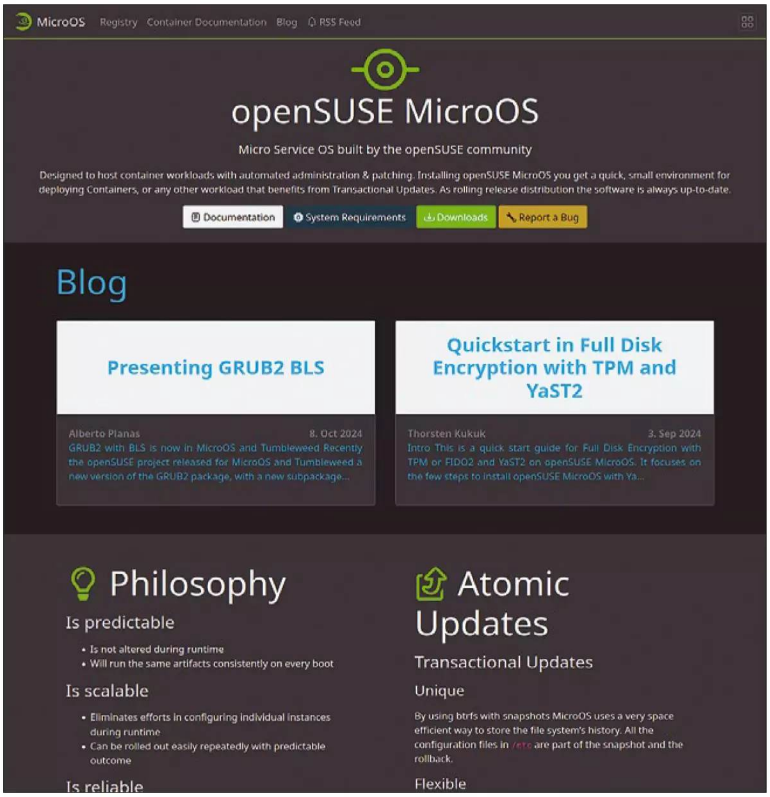
and log in as this user. Throughout this article, all containers are operated by this user without the need for root privileges.

Configuring Container Registries

When pulling a container image, you often do not need to specify the container registry because Podman automatically searches a few trusted registries that are configured in the `/etc/containers/registries.conf` file. On an openSUSE MicroOS system, for example, this file includes:

```
unqualified-search-registries = ⮐
  ["registry.opensuse.org", ⮐
   "registry.suse.com", "docker.io"]
```

This sequence implies Podman first searches `registry.opensuse.org`, followed by `registry.suse.com`, and finally `docker.io`. If this configuration file exists at `~/.config/containers/registries.conf`, it takes precedence for that user over the system-wide configuration file in `/etc/containers/registries.conf`.



openSUSE MicroOS

Micro Service OS built by the openSUSE community

Designed to host container workloads with automated administration & patching. Installing openSUSE MicroOS you get a quick, small environment for deploying Containers, or any other workload that benefits from Transactional Updates. As rolling release distribution the software is always up-to-date.

Blog

Presenting GRUB2 BLS

Alberto Planas 8. Oct 2024

GRUB2 with BLS is now in MicroOS and Tumbleweed Recently the openSUSE project released for MicroOS and Tumbleweed a new version of the GRUB2 package, with a new subpackage...

Quickstart in Full Disk Encryption with TPM and YaST2

Thorsten Kukuk 3. Sep 2024

Intro This is a quick start guide for Full Disk Encryption with TPM or FIDO2 and YaST2 on openSUSE MicroOS. It focuses on the few steps to install openSUSE MicroOS with Ya...

Philosophy

Is predictable

- Is not altered during runtime
- Will run the same artifacts consistently on every boot

Is scalable

- Eliminates efforts in configuring individual instances during runtime
- Can be rolled out easily repeatedly with predictable outcome

Is reliable

Atomic Updates

Transactional Updates

Unique

By using btrfs with snapshots MicroOS uses a very space efficient way to store the file system's history. All the configuration files in /etc are part of the snapshot and the rollback.

Flexible

Figure 1: An immutable operating system such as openSUSE MicroOS is an ideal base OS for running Podman containers.

Allowing Unprivileged Ports

Operating a web server on a standard port like 80 (HTTP) or 443 (HTTPS) is often preferred, because visitors then don't need to specify a port number in the URL. By default, though, unprivileged processes – which includes rootless Podman containers – are only allowed to listen on ports above 1024. This port restriction can be easily adjusted with a kernel parameter by creating the configuration file `/etc/sysctl.d/10-rootless-podman.conf`, decreasing the lowest port unprivileged processes can listen on to 80:

```
net.ipv4.ip_unprivileged_port_start=80
```

Then load the new kernel parameters:

```
$ sudo sysctl --load /etc/sysctl.d/⮐
  10-rootless-podman.conf
```

Afterward, your rootless Podman container with a web server can listen on port 80 on the host. From the next reboot, this file will be loaded automatically.

User Permissions

Users inside a rootless container must be mapped to a user on the host. To do this correctly, you need to allocate subIDs to your user with a command like,

```
$ sudo usermod ⮐
  --add-subuids 100000-65536 ⮐
  --add-subgids 100000-65536 koan
```

which allocates 65,536 sub-IDs starting from 100,000 to the user `koan` (substitute your username). As a result, the `/etc/subuid` and `/etc/subgid` files must both contain the following entry for your user:

```
koan:100000:65536
```

If multiple users need to run rootless Podman containers, repeat this task for each user. Ensure that the sub-IDs don't overlap. For example, have the second user start from sub-ID 165,536.

Automatic Startup

You still need to prepare one more thing before you can start defining systemd unit files for your containers: automatic startup. The complication of working with rootless Podman containers means that, by default, user services only start when the user is logged in and stop when the user logs out. This arrangement is inconvenient for server applications in rootless

Podman containers, because the user isn't always logged in to the server. Moreover, if the server reboots (e.g., after a power outage), the user is no longer logged in, and the containers don't start. You want to avoid such a situation.

Listing 1: `freshrss.container`

```
01 [Unit]
02 Description=FreshRSS feed aggregator
03
04 [Container]
05 Image=docker.io/freshrss/freshrss:1.25.0
06 ContainerName=freshrss
07 HostName=freshrss
08 Volume=%h/containers/freshrss/data:/var/www/FreshRSS/
     data:Z
09 Volume=%h/containers/freshrss/extensions:/var/www/
     FreshRSS/extensions:Z
10 Environment=TZ=Europe/Brussels
11 Environment=CRON_MIN=1,31
12 PublishPort=80:80
13
14 [Service]
15 Restart=always
16
17 [Install]
18 WantedBy=default.target
```

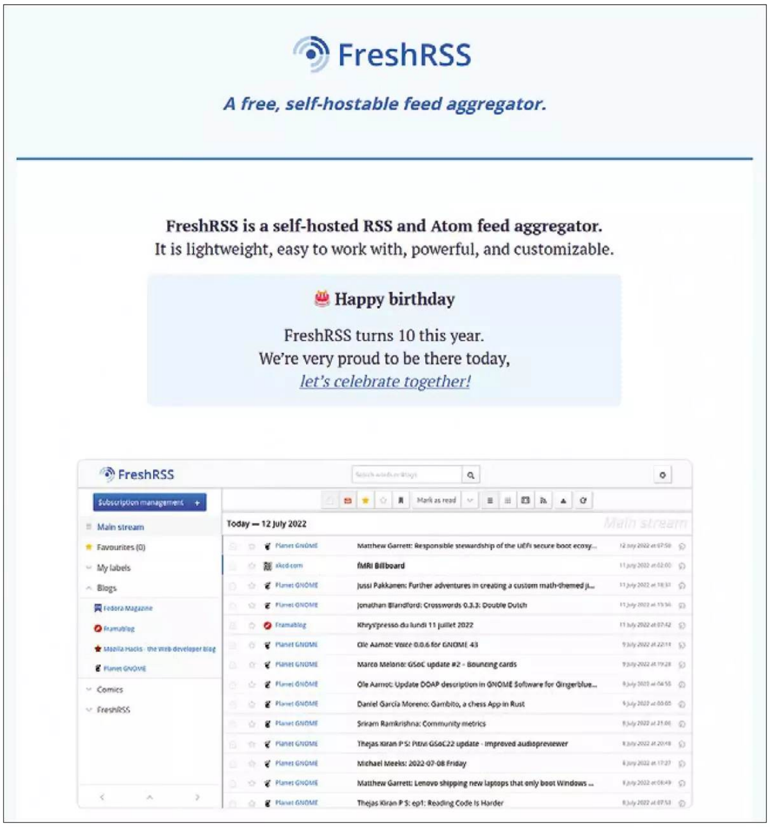
Fortunately, this behavior can be altered easily by running the command

```
$ loginctl enable-linger
```

under the relevant user account. Systemd will now start a user manager for your user at your distribution's boot time, which remains active and ensures that your user services for your containers start as configured. The `linger` indicates that your processes persist after user logout.

Defining a First Container

After all this preparation, you can now define your first systemd unit file for a container. As an example, set up FreshRSS [7], a self-hosted RSS feed aggregator (Figure 2). To have this service start automatically in a Podman container, define the systemd unit shown in Listing 1 in the file `~/.config/containers/systemd/freshrss.container`.



FreshRSS

A free, self-hostable feed aggregator.

FreshRSS is a self-hosted RSS and Atom feed aggregator.
It is lightweight, easy to work with, powerful, and customizable.

Happy birthday

FreshRSS turns 10 this year.
We're very proud to be there today,
let's celebrate together!

Figure 2: FreshRSS is a self-hosted RSS and Atom feed aggregator, which runs perfectly in a rootless Podman container.

If you are familiar with systemd unit files for services, you'll notice a new section: `[Container]`. The file name also uses the `.container` extension instead of `.service` to indicate that it's a container unit.

The `[Container]` section has various options for your container, documented extensively in the Podman Quadlet [4] man page. In this example, I define the container image, name, hostname, volumes, environment variables, and ports you want to publish.

The `Volume` statement in line 8 indicates that the directory `containers/freshrss/data` in the host's home directory (`%h`) of the user is mounted at `/var/www/FreshRSS/data` within the container. On an SELinux-enabled system, the `Z` option specifies that this directory is used only by this container, ensuring it receives the proper labeling. If the directory is shared with other containers, use the lowercase `z` option instead, and use `ro` for a read-only mount within the container.

Under the user's home directory, create these directories on your host system:

```
$ mkdir -p ~/containers/freshrss/➘
           {data,extensions}
```

I typically create a `~/containers` directory within the user's home directory with a subdirectory for each rootless container. If you prefer a different setup, adjust the volumes in the container unit file accordingly.

Next, the `[Container]` section of the unit file has a line for each environment variable and then a line to publish the container's port (line 12). In this case, container port 80 (defined after the colon) from the container is exposed to port 80 on the host (defined before the colon).

The `[Service]` and `[Install]` sections of the unit file are familiar from service unit files. The `[Install]` section in particular is important: The `WantedBy=default.target` option ensures the container starts on boot.

Starting the Service

After saving the container file in Listing 1 and creating the directories for the container's volumes, you need to inform systemd of the new unit file for the user:

```
$ systemctl --user daemon-reload
```

This command generates a service for the container in the background, with the same name as your `.container` unit file but with the `.service` extension. You can view its content with:

```
$ systemctl --user cat freshrss.service
```

You'll see, for instance, that the service unit file contains an extensive `[Service]` section with the `ExecStart` option consisting of a `podman run` command that starts the container with various options like volumes, environment variables, and so on.

You can now also start the container by starting this generated service:

```
$ systemctl --user start freshrss.service
```

The startup might take some time if the container image needs to be downloaded first.

If all goes well, the FreshRSS web interface is available on port 80 of your container host. If not, consult the logfiles with `journalctl` or take a look at the output of:

```
systemctl --user status freshrss.service
```

Always remember to use the `--user` flag in your `systemctl` commands; because you're working with rootless containers, the corresponding systemd services are running with user privileges.

Creating a Podman Network

You can similarly define unit files for volumes, networks, pods, and more. As an example, I'll define a Podman network in a systemd unit file. Instead of letting all your containers publish their ports directly on the host, connect them all to a Podman network and then run a container with a reverse proxy to access all those containers externally.

Define a `.config/containers/systemd/reverse-proxy.network` file in your home directory with the content:

```
[Unit]
Description=Reverse proxy network

[Network]
NetworkName=reverse-proxy
```

Additional options are documented in the aforementioned Podman Quadlet man page. Now change the unit file of your FreshRSS container to include the following line in the `[Container]` section to use this network:

```
Network=reverse-proxy.network
```

Remove the `PublishPort` line because the container's port shouldn't be exposed on the host anymore: That will be the reverse proxy's task.

Reverse Proxy Preparation

Before you create a container with Caddy [8] as a reverse proxy, first take a step back and review some prerequisites. The goal is to access each of the containers on its own subdomain of a domain you own. The reverse proxy will listen to all subdomains and forward HTTP requests to the corresponding container's hostname and port on the basis of the visited subdomain.

By default, Caddy obtains and renews TLS certificates automatically with Let's Encrypt [9], which is perfect for publicly accessible services. However, if you're running services on a local network and no one else should access them, it makes sense to create a wildcard certificate manually for your local domain, signed by your own Certificate Authority (CA). You then also define a wildcard domain in your local DNS server, such as `*.services.home`, pointing to your container host's IP address and use this wildcard domain in your certificate.

I did both tasks with OPNsense [10], the software running on my router, because I'm using this CA for other hosts in my internal network, too. Alternatively, Caddy can generate a CA and sign certificates automatically for internal use. The details go beyond the scope of this article. You should change Caddy's configuration to suit your setup.

Caddy Reverse Proxy

Now, create the systemd unit file `~/.config/containers/systemd/caddy.container` with the content in Listing 2. This code makes Caddy listen on port 443 for HTTPS and connects it to the reverse proxy network, as well, so it can reach the FreshRSS container by way of its hostname.

It also defines some volumes. Note that the volume for the certificates is read-only and shared with other containers (`z,ro`).

Next, create the directory `~/containers/certificates` and add the certificate and key file for the wildcard domain. Also, create the directories `~/containers/caddy/data` and `~/containers/caddy/etc`. Within the `/caddy/etc` directory, you should create a configuration file named `Caddyfile` and give it the content:

```
*.services.home {↩
  tls /data/certificates/services.crt/↩
  data/certificates/services.key
}
```

Listing 2: `caddy.container`

```
[Unit]
Description=Caddy reverse proxy

[Container]
Image=docker.io/library/caddy:2.8.4
ContainerName=caddy
HostName=caddy
Volume=%h/containers/caddy/data/caddy:/data/caddy:Z
Volume=%h/containers/certificates:/data/certificates:z,ro
Volume=%h/containers/caddy/etc:/etc/caddy:Z
PublishPort=443:443
Network=reverse-proxy.network

[Service]
Restart=always

[Install]
WantedBy=default.target
```

```
freshrss.services.home {
  reverse_proxy freshrss
}
```

This code uses the TLS certificate you created with common name `*.services.home` for all your subdomains of `services.home`. If `freshrss.services.home` is requested, Caddy forwards the request to the `freshrss` hostname on port 80.

Now reload the systemd daemon to make it aware of this new file and the changed `freshrss` container unit file and (re)start the services for both containers:

```
$ systemctl --user daemon-reload
$ systemctl --user restart freshrss
$ systemctl --user start caddy
```

If all goes well, you should be able to visit FreshRSS in your web browser on *https://freshrss.services.home*. If not, have a look at the logs of Caddy's Podman container.

This setup can be readily expanded by adding new unit files for additional containers and appending entries in Caddy's configuration file to forward requests to the appropriate container. If a container listens on a port other than 80, just add the port to the `reverse_proxy` line in the Caddyfile (e.g., `reverse_proxy homepage:3000`). You can even add an authentication portal to your reverse proxy with the *caddy-security* plugin.

Conclusion

Rootless Podman containers offer a secure method to operate services on a Linux system, which, combined with systemd unit files, makes it exceptionally flexible to configure. For instance, if one container depends on another, you can specify this arrangement with `Requires=` and `After=` in the `[Unit]` section, just as you would for systemd services. Note that you need to mention the name of the generated service with the `.service` extension. All other options in systemd unit files in the `[Unit]` and `[Service]` sections are supported, as well.

Additionally, you can simply query the status of a container with

```
systemctl status <container-name>.service
```

Upgrading a container is merely a matter of changing the version number in the container unit file, reloading the systemd daemon, and restarting the service. Personally, I transitioned an installation with Docker Compose on my home server to the approach outlined in this article some time ago, and I consider it a significant improvement. ■

Info

[1] **Docker Compose:** [https://docs.docker.com/compose/]
[2] **Podman:** [https://podman.io]
[3] **Podman Compose:** [https://github.com/containers/podman-compose]
[4] **Podman Quadlet:** [https://docs.podman.io/en/latest/markdown/podman-systemd.unit.5.html]
[5] **Fedora CoreOS:** [https://fedoraproject.org/coreos/]
[6] **openSUSE MicroOS:** [https://microos.opensuse.org]
[7] **FreshRSS:** [https://freshrss.org]
[8] **Caddy:** [https://caddyserver.com]
[9] **Let's Encrypt:** [https://letsencrypt.org]
[10] **OPNsense:** [https://opnsense.org]

The Author

Koen Vervloesem has been writing about Linux and open source, computer security, privacy, programming, artificial intelligence, and the Internet of Things for more than 20 years. He holds Master's degrees of Engineering in Computer Science and Philosophy and teaches Linux, Python, and IoT classes. You can find more on his website at [koen.vervloesem.eu].

■

Hone Your Skills
– with –
Special Issues!

Get to know Shell, LibreOffice, Linux, and more from our Special Issues library.

The *Linux Magazine* team has created a series of single volumes that give you a deep-dive into the topics you want.

Available in print or digital format

background Image © roystudio, 123RF.com

Check out the full library!

shop.linuxnewmedia.com

Web applications with Julia

On the Rise

The technical programming language Julia is fast becoming a favorite with scientists and engineers of all stripes, but it is also well suited to form the backbone of interactive websites that explain these science and engineering concepts. By Lee Phillips

Julia [1] is a high-level, high-performance, general-purpose programming language. The "high level" refers to its expressive syntax, advanced data structures, and sophisticated type system, which are features that allow the programmer to solve complex problems with compact, readable programs [2]. High performance is achieved with a unique just-in-time compilation system that turns high-level code into native machine instructions on any of Julia's many supported architectures, including GPUs. Julia is a general-purpose language because it's convenient to use in any problem domain. The only areas in which Julia might not be an ideal choice, because of the overhead of compiling, are embedded and real-time applications (although even this may change [3] in the future).

Julia was originally envisioned as a new language [4] for scientific and technical computing. Since its relatively recent public release in 2012, its unique melding of advanced concepts in language design with uncompromised performance has led to rapid and enthusiastic adoption across many science and engineering disciplines. It's one of only four languages [5] that have been deployed on supercomputers for petaflop computations (the others are Fortran, C, and C++).

Because of the emphasis on technical computing in the Julia community, most of the books [6] and online documentation and tutorials are aimed at scientists and engineers who are interested in using the language for simulation, data analysis, and visualization of results; this is certainly true of my own book [7], for example. However, Julia is an excellent choice for other types of applications as well.

Julia on the Web?

Recently I've used Julia as a back-end language for interactive websites to demonstrate concepts in physics. Because very little information was available explaining how to build such projects, I had to assemble the required knowledge from disparate sources and complete the picture with a great deal of experimentation and testing. This article is the culmination of this process and represents my desire to present, in one place, a complete guide to creating interactive web applications with Julia.

I found that Julia was an ideal language for this purpose. All of the virtues that make it so well suited for computational tasks in science make it delightful to use on the back end of a web application, especially one involving physics calculations. In addition to the features already mentioned, Julia permits the use of Greek letters and Unicode symbols in variable names, which lets the scientist-programmer impart a familiar appearance to lines of code, making them more closely resemble his or her beloved equations. Julia's wide collection of graphics and audio libraries present convenient solutions for the task of constructing diagrams, animations, or sounds to illustrate the science concepts that I seek to explain in my web pages.

It's also almost trivially easy in Julia to spawn background tasks or perform calculations in parallel on multiple processors, which are factors that can improve responsiveness over the web. I discovered that Julia has excellent libraries for network communication, most critically for talking over WebSockets, which turned out to be a key element for my task. The parts of my programs for handling communication with the browser were surprisingly concise and easy to write, and the performance was excellent, which finally convinced me that the experiment was a success and made the techniques described here my favorite methods for building these types of interactive sites.

In this article, I document two methods for creating interactive web applications with Julia. The first uses the PlutoSliderServer web server [8], which allows you to transform a Pluto [9] notebook into a web application that you can safely make available on the public web. You can use this method even if you're unfamiliar with HTML or JavaScript, because you can build your application from a large

Lead Image © sony sivanandan, 123RF.com

collection of pre-made widgets that automatically set the values of variables in your Julia programs. The second method is more open-ended and allows you to build any type of web application that you can imagine. It's based on HTMX **[10]** and some simple (optional) JavaScript on the front end, communicating over WebSockets with your back-end Julia programs.

I'll explain all the terms and ideas in this paragraph in the following sections, so don't be alarmed if I've mentioned some unfamiliar things. They're all well worth learning.

PlutoSliderServer

Pluto is a computational notebook that uses the web browser as an interface. It's an evolution of the popular Jupyter **[11]** notebook, improving on the concept by fixing **[12]** several problems in Jupyter's design. Whereas Jupyter can serve as an interface to many different languages (the name is a mashup of Julia, Python, and R, its original partners), Pluto works, and will likely forever only work, with Julia.

Like Jupyter, Pluto was originally intended to provide a convenient interface to a language kernel running locally on the same machine as the web browser that handles the Pluto interface. Although some solutions allow Pluto to work over the web, I don't want, for my purposes here, simply to expose a notebook to the user.

Ideally I want a simple means to translate a Pluto notebook to a web page, with interactivity provided by widgets with well-defined behavior; I don't want the visitors to the page to be able to execute arbitrary Julia code, which is what the full Pluto interface enables.

PlutoSliderServer offers the perfect solution. With this approach, you create a Pluto notebook that includes any computations, media, and explanatory prose (with Markdown or HTML) you desire. User interactivity is provided by a large collection of widgets (including the popular slider control, whence the name) from the *PlutoUI* **[13]** package. Each widget is bound to the value of a variable, and when the user manipulates the control to change the value of the variable, the relevant code is immediately executed on the server with the new value, displaying the results.

Here, I'll go through every step required to create and serve a web application with this approach. The first section of **Table 1** gives you an idea of the kinds of things you can make with PlutoSliderServer. The URLs lead you to two pages I made with this technology, as well as a link to an MIT intro course on computational thinking, which uses PlutoSliderServer extensively in its online lectures.

Installation and Setup

I'll explain everything you need on this journey, but the trip will be a little smoother if you have just a little familiarity with Julia and its package system **[14]**.

The first step is to install Julia if you don't have it already. I recommend an upgrade if your installed version is older than version 1.8. I tested all examples in this article with version 1.10; version 1.11 recently became available, but I encountered some issues with package compatibility under that version with at least one of the examples, so version 1.10 might be smoother sailing, at least until the problems with Pluto under the most recent release have been resolved.

To install Julia on any operating system in common use, visit the Julia website and click the big green *Download* button. Once you have Julia up and running, you will need a few packages. A "package" in Julialand is a library of functions and data structures that you can import and use in your programs.

For developing a PlutoSliderServer application, you'll need the *Pluto* package. Start up the Julia REPL (read-eval-print loop interactive terminal prompt) by typing `julia` in a terminal window. Of course, you can also use Visual Studio (VS) Code or your favorite development environment, and the procedure will be basically the same. Although you can now continue your development of PlutoSliderServer web applications in your default Julia environment, it's a better practice to create a separate environment for each project to help prevent package conflicts down the road and keep your work organized. An "environment" in Julia is simply a directory in your filesystem containing two special files Julia maintains that list dependency-resolved packages needed in your project, along with all your program files. The latter are text files with the `.jl` file extension.

After deciding where in the filesystem you want your new project to live, press the right square bracket (*]*) key in REPL to activate package mode. The prompt changes from the green *julia >* to a blue *pkg >*. This new prompt is preceded by a notation in parenthesis indicating your current

Table 1: Example Applications

Example	Source
PlutoSliderServer Examples	
A demonstration of an assortment of widgets	[https://pluton.lee-phillips.org/sliders/uiDemo.html]
Ptolemy's universe	[https://pluton.lee-phillips.org/sliders/epicycles.html]
Introduction to computational thinking. A. Edelman, D.P. Sanders, and C.E. Leiserson, lecturers. MIT	[https://computationalthinking.mit.edu/Fall24/]
HTMX Examples	
A simulation of the acoustics of plucked string instruments	[https://lee-phillips.org/pluckit]
A simulation of two-dimensional vortex dynamics	[https://lee-phillips.org/vortex]

environment (Figure 1). Initially you're in the default environment for the version of Julia in use (in this example, version 1.10).

The next step is to activate the new environment you've settled on for your project; the command for this is also shown in the figure. At this point if you press the backspace key you'll return to normal REPL mode. However, you want to remain in package mode a little longer to add the one package needed for all Pluto development: the *Pluto* package. In package mode, at the *pkg >* prompt, enter the command

```
add Pluto
```

to download and precompile the package's program files (you need to be connected to the Internet). The process can take anywhere from less than a minute to several minutes, depending on the speed of your network connection and of your computer.

Pluto notebooks contain their own environments wholly encapsulated within the notebooks. This arrangement helps make them portable and their calculations reproducible, which is an important consideration for the scientists who comprise a good proportion of their user community. Therefore, any further packages needed for calculations in the notebook will not be added in the REPL but will be imported in the notebook. Julia and Pluto take care of these packages behind the scenes.

Developing a Notebook

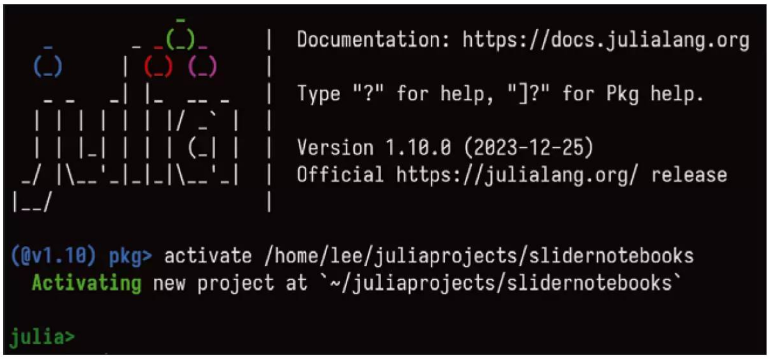
You will be developing your notebook within the web interface provided by Pluto. The next step in starting up this interface, the connection between the web browser and the Julia process running on your computer, is to exit package mode in the REPL and, at the normal green *julia >* prompt, execute the commands:

```
using Pluto
Pluto.run()
```

After some messages in the REPL and a short delay, a new window opens in your default web browser displaying the Pluto interface (Figure 2).

To replace this start page with an empty Pluto notebook, click the control that says *Create a new notebook*, where you'll begin the development of your web application. The first step, to remain organized, should be to click on the *Save notebook* text near the top of the page to give the notebook a name and a place in the filesystem. Now you can put any code in any of the notebook cells; rearrange the cells to present the notebook in a logical, pedagogically useful manner; and click on the little eye icons at the left of focused cells to decide whether your users will see their content or merely their results. If you haven't used Pluto or Jupyter (which follows similar conventions) before, it might be helpful at this point to consult one of the relevant articles or books linked elsewhere in this article; however, the interface and behavior is fairly intuitive.

If you have used Jupyter but have no experience with Pluto, you must be aware of one crucial difference: In Jupyter, cells are executed from the top down, but in Pluto, their order on the page has no effect on their execution. Cells in Pluto are executed in a logical order, following the graph of their dependencies. If you cause any cell to be executed, all the cells whose results depend on the outcome of that cell are automatically executed as well, following the chain of dependencies until everything is resolved. Therefore, you can order the cells to make the most presentational sense and even change your mind and the order later, without affecting the results. This freedom of cell ordering carries through to the web application

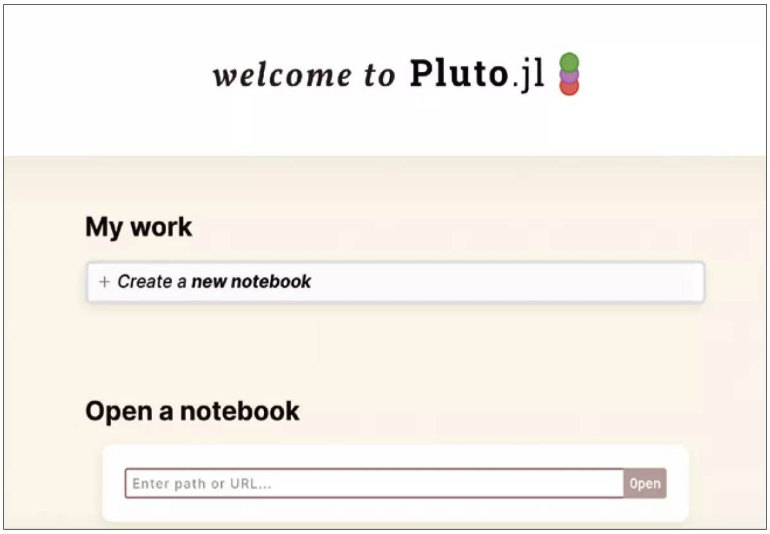


```
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.10.0 (2023-12-25)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

(@v1.10) pkg> activate /home/lee/juliaprojects/slidernotebooks
  Activating new project at `~/juliaprojects/slidernotebooks`

julia>
```

Figure 1: Activating an environment in the Julia REPL in package mode.



welcome to Pluto.jl

My work

Open a notebook

Figure 2: The Pluto developer's starting page.

that your notebook becomes: The user will be able to manipulate the controls in any order and will always see a reproducible, correct result.

You can use almost any package in the notebook by simply importing it with a `using` statement in any cell; the consequent downloads and compilations happen automatically. The one required package for any notebook destined for PlutoSliderServer is *PlutoUI*, which contains the code for the widgets with which users interact. In addition to the familiar slider, PlutoUI provides a large and useful collection of widgets [15] with which you can design a huge variety of interactive experiences.

Each widget performs the same function: It sets, or binds, the value of a variable to a value determined from the user's manipulation of the control. To insert a widget on the page, use the `bind` macro, supplying it with a first argument naming the variable to bind and a second argument describing the widget to use. Every time the user manipulates one of these widgets, the value of its associated variable is (potentially) changed, and any cell that depends on the value of that variable is run (followed by all cells that depend on the result of that cell). This technique is thus ideal for demonstrations of such things as showing how the solution to an equation depends on the value of a parameter.

Figure 3 illustrates the use of the `bind` macro to create widgets. The notebook first does the necessary import in the first cell (this would normally be hidden from end users of the application). The second cell uses Julia's macro syntax to invoke the macro to bind the value of a slider (which can take integer values from 1 to 10) to the variable `x`. In the figure it's set at the value 6 with the slider. The third cell shows the use of the `NumberField` macro, with a keyword argument supplying its default value. Note that in Pluto a cell's result is displayed above the cell content.

The final cell uses a Markdown "nonstandard string literal" to represent a Markdown string. The definition of this string literal is included neither in Julia nor in Pluto (nor in PlutoUI), but in another package that is automatically imported in all Pluto notebooks because the use of Markdown to create text cells is so common. The string has interpolated variables and uses Markdown syntax for italics and boldface. You can also use HTML if Markdown doesn't provide everything you need, either with an HTML nonstandard literal (`html"<markup goes here>"`) or with the `HTML()` function, which is able to incorporate interpolated variables that don't work in an HTML literal. An even more flexible way to use HTML in Pluto is with the *HypertextLiteral* package [16], which I use extensively in my Ptolemy's Universe example.

When this notebook is made available over the Internet by PlutoSliderServer, the user will be able to use the slider and the input box to set the values of `x` and `y`, but not be able to change the code in the cell that computes their sum. Whether that code is visible or not is your choice, depending on your purpose for the notebook.

Setting Up the Server

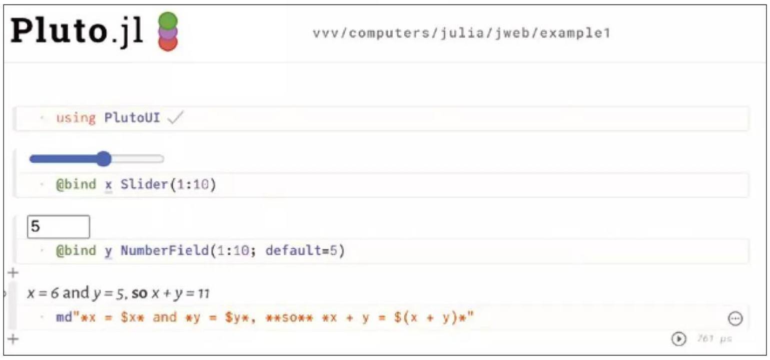
You are going to serve a web page on the public Internet that allows visitors to run Julia programs on a remote server. Although the interactions through PlutoSliderServer are deliberately constrained, the possibility is always present that a user will find a way, either maliciously or inadvertently, to do something that is not part of your plans (e.g., the Julia process running on the server potentially has access to the filesystem). This software comes with no security guarantees; it's up to the developer to take measures to limit the scope of potential damage.

The best, and recommended, way of accomplishing this added security is to confine the Julia process on the server to some type of virtual environment, such as a container from which it is impossible to see the machine's root filesystem. The use of such a container makes the worst case scenario the destruction of the container itself; the server machine will be unaffected, and the administrator can start up the container again.

Among the several widely used container strategies is my favorite, which is built into most Linux systems: `systemd-nspawn`. If you're using a Linux server and don't already have a favorite container solution of your own, I suggest learning how to use `systemd-nspawn` to secure and isolate the server environment [17]. Another popular option you might already know about is Docker. It's not important what container technology you use, but it is important that you do something to isolate the Julia process on your server.

When merely testing out the notebook with the slider server, you can of course do everything on your local machine, where no container will be necessary.

After setting up your container and installing Julia on it, you should start a Julia REPL on the container, establish an environment as above (or simply use the default environment), and import the *PlutoSliderServer* package, which pulls in Pluto as a dependency.



```
using PlutoUI
```

```
@bind x Slider(1:10)
```

5

```
@bind y NumberField(1:10; default=5)
```

$x = 6$ and $y = 5$, **so** $x + y = 11$

```
md"*x = $x* and *y = $y*, **so** *x + y = $(x + y)*"
```

Figure 3: A Pluto notebook with PlutoUI widgets.

The next step is to configure the web server on the host machine (not within the container). The following instructions will be specific to Apache, because that's where I tested my solutions. The configuration for NGINX will be quite similar.

The goal is to configure Apache to act as a gateway, or reverse proxy, to intercept normal HTTPS web requests to your public server and relay them to the PlutoSliderServer web server running in your container. Your visitors will not have to know anything about the slider server; they'll interact with your web application just as with any normal web page.

For Apache, first install or enable the required modules to enable proxying: the `proxy`, `proxy_html`, and `proxy_http` modules. These might already be installed; check in the `/etc/apache2/mods-enabled` directory. Next, add the lines in **Listing 1** to the appropriate section of the configuration file for your virtual domain, if you have one, or for the default. For Apache, the files live in `/etc/apache2/sites-enabled/`.

The listing shows the proxy configuration for my slider pages, where I use `/sliders/` in the URLs. Adjust those lines as needed for your server configuration. The directives pass requests from the user's browser to port 2345 on the local machine, which is the default port used by the *PlutoSliderServer* package. You can change the port if you happen to be using that one for something else (see below). The 127.0.0.1 address refers to the local machine and is how you can talk to the slider server running on the container. Note that communication within the server machine is over `http` – hence, in plain text. Although OK for now, you should set up Apache (NGINX, etc.) so that communication over the Internet uses `https`.

The final step is to write some small scripts to start Julia and PlutoSliderServer in the container. You should be logged in as root within the container to complete this final step.

Listing 1: Apache Gateway Config

```
ProxyRequests Off
AllowEncodedSlashes On
ProxyPass "/sliders/" "http://127.0.0.1:2345/" nocanon
ProxyPassReverse "/sliders/" "http://127.0.0.1:2345/"
  nocanon
```

Just running Julia from the terminal won't quite work, because the process exits when you log out of the container. Even using `nohup` is not a reliable way to keep Julia running. For this purpose, use the `systemd-run` command, which, like `systemd-nspawn`, is built in to most modern Linux distributions (at least those that use systemd, which, although far from beloved by all, has become a standard).

Specifically, create a file with the startup script:

```
#! /usr/bin/bash
echo "Notebook server starting"
julia
echo "Done."
```

Name it (say) `servenotebooks`, and make it executable with:

```
chmod 755 servenotebooks
```

As you can see, all the script does is start Julia. Invoking it with `systemd-run` makes the Julia process persistent; that's its entire reason for existing.

One wrinkle might pop up that you will have to iron out, depending on the packages you use in your notebook. Some packages assume that Julia is being run on a computer with a graphical display (e.g., the X Window System); however, typically, this is not the case when running Julia on a server. It happens to be a problem with one of my example notebooks, which uses the *Javis* [18] package for creating animations. Javis will not even compile unless it detects a system for graphical output.

One solution to this problem, if you encounter it, is to install X Windows on the server. Another solution is to install the program `xvfb-run`, which creates a virtual X Window environment for a program to run within. To use this solution, simply replace `julia` in the startup script above with `xvfb-run julia`.

When Julia starts, it runs the `.julia/config/startup.jl` Julia script in your home directory. Because Julia will start in this container only for the purpose of running PlutoSliderServer, you can handle it in the startup script. Create (or edit, if it already exists) a file with the path mentioned here, providing it with the content:

```
using PlutoSliderServer
run_directory(
  "/<the/notebook/directory>/";
  Export_offer_binder=false)
```

The first line imports the only package needed to serve the notebooks; the package was added in a previous step. The second line starts the slider server in a mode where it monitors the directory in which you want to put your notebooks (replace `/<the/notebook/directory>/` with this location). In this mode you can add, delete, or upload updated versions of all your notebooks, and the slider server will take note and serve the current state of the directory. You won't need to restart the server or take any other action.

The keyword argument (the bit after the semicolon) to `run_directory` tells the server not to put a particular button on the notebooks you don't want. To see a list of all available options for this function, enter help mode in the Julia REPL by typing *? <function name>*.

With this step completed, your server is ready and running. You can check its status with the `journalctl` command or kill it with the `systemctl kill` command. Use the

```
julia --startup-file=no
```

command if you ever want to start Julia in your container without executing the startup script.

HTMX Web Applications

PlutoSliderServer is ideal if you already have a Pluto notebook that you want to turn into a web application or if the application you have in mind fits well with the PlutoUI set of widgets and the notebook framework.

In this section, I describe another way to create interactive web applications that use Julia on the back end for computations. With these methods, you don't need Pluto or any other Julia package aside from those used for your calculations and the *HTTP* [19] package, which provides functions for communication over a WebSocket.

The front-end interactions are designed around HTMX, which is a small JavaScript library that enhances HTML by allowing more elements to send messages to the server and by easily allowing fragments of the page to be replaced with the server's response, rather than having to reload the entire page on each request (similar to the AJAX method of request-and-response handling, but extended and made easier to use, with no explicit JavaScript required). This enhancement is exactly what's needed for a smooth interactive experience.

The method also uses the WebSocket extension to HTMX [20]. PlutoSliderServer uses WebSockets to make interactions snappier, although you didn't have to be aware of that when setting up your client and server.

The method described in this section also uses WebSockets for the same reason.

Visit the sites under the HTMX Examples section in Table 1 for two examples of web applications I made with the methods from this section. These examples give you some idea of the kinds of things you can make, although your imagination is the only limitation.

You'll need to import the small HTMX library and its WebSocket extension into your HTML page. You can import directly from their source URLs provided in the documentation, but I prefer to download them to my server and load them locally to avoid potential problems, including when accessing the HTMX website. If the visitor can get to your page, then the HTMX library is guaranteed to load. The drawback is losing the potential advantage of the user's browser cache on first visit, but this is not a huge consideration because these JavaScript libraries are quite compact.

Put something like the following two lines near the top of the HTML page of your web application:

```
<script src="/htmx.js"></script>
<script src="/ws.js"></script>
```

This example is for when you've placed the files at the root of your web directory, but of course you can put them elsewhere. The second line loads the WebSocket extension.

Because you won't have access to the PlutoUI collection of widgets, you'll need to have some familiarity with HTML to be effective with the method described here. Additionally, it's useful to know a little bit of plain JavaScript, in case you'd like to customize the user interactions beyond what HTMX and HTML can provide on their own (although you can go pretty far with these two technologies without the need for any JavaScript at all).

The next example illustrates the differences in defining widgets between PlutoUI and HTML. The previous examples created a slider in the Pluto notebook (Figure 3) with

```
@bind x Slider(1:10)
```

In HTML, the closest equivalent is:

```
<input type="range" name="s"
       min="1" max="10">
```

However, the two examples already show a big difference in behavior. The `@bind` macro invocation in the Pluto notebook sets the value of `x` directly when the user manipulates the slider, and any notebook cells that depend on the value of that variable are immediately executed (as well as all those further down in the chain of dependencies). All you need to do is write the programs that use `x` and put in the `@bind` macro for the slider anywhere it makes sense in your presentation.

In the HTML case, the `<input>` tag merely puts a slider on the page. You have to do a bit more to make anything happen when the user slides the control. Listing 2 shows a fragment of an HTML page for a minimal slider that adds three attributes to the input tag for the slider (the tag is broken over three lines here).

All of these attributes invoke functions from the HTMX WebSocket extension (see the "Validation and the Data Prefix" box). The new attributes in Listing 2 specify, respectively, that the element (the slider) uses the extension, that it sends data over a websocket, and that it will send the data to the specified address. The WSS protocol is for encrypted WebSocket communication (analogous to HTTPS), which you should always use when communicating over the public Internet. For testing on your local machine, you can replace the address with

```
ws://127.0.0.1:PORT
```

and use the port number that you've set up in place of `PORT`. Finally, the element is wrapped in a label, which associates the instruction with the slider. Following the labeled slider is an empty paragraph with an `id` that is there to receive the response from the server.

Every time the user changes the position of the slider, a value associated with *name s* is packaged into a message and transmitted over the open WebSocket to Julia running on the server. The client sends the message after the user stops interacting with the control (e.g., by moving the mouse away), so you needn't worry about a massive cascade of messages being transmitted while the slider is in motion.

Listing 3 shows a minimal working Julia program that defines a server that can respond to the slider in the previous example. To begin, create an environment for your Julia project in the REPL's package mode, as before, and add the *HTTP* and *JSON* packages.

Listing 2: Minimal Slider

```
<label> Pick a number
  <input data-hx-ext="ws" data-ws-send
    data-ws-connect="wss://<your/server/url>/"
    type="range" name="s" min="1" max="10">
  </label>
<p id='sr'></p>
```

Validation and the Data Prefix

The `data` prefix in the three new attributes in the `<label>` tag of Listing 2 is optional, in the sense that its omission will not affect the behavior of the client. It is there to make the page valid HTML; custom attributes such as `hx-ext` are not part of HTML, so validators will complain; however, they will ignore any attribute beginning with `data-`. Allowing the `data` prefix is a feature of HTMX. Note that PlutoSliderServer pages do *not* pass validation, and their sources cannot be usefully inspected by a curious user, which is a disadvantage. Pages made with HTMX can be valid HTML, and their code is legible with `view source`.

The JSON package is here for one function: `JSON.parse()`, which extracts variables from the messages sent by the client. The information returned by the server is always in the form of HTML fragments, not JSON: This is the philosophy behind HTMX.

Variables extracted from browser messages are always strings. In this example, the string is passed to `Meta.parse()`, part of Julia Base, which converts strings to Julia expressions; in this case, it's used to convert the string to an `Int`.

The HTTP package provides the two communication functions around which you will build your applications. `WebSockets.listen()` takes a second argument giving the IP number on which to listen, and a third argument for the listening port. This function opens a WebSocket connection on the given address and port. The address shown in the listing refers to the local machine, either your development computer or the server container on which you'll eventually deploy.

Listing 3: Minimal Server Example

```
using HTTP, JSON
const PORT = 8660

function startserver()
  WebSockets.listen("127.0.0.1", PORT) do ws
    for msg in ws
      d = Meta.parse(JSON.parse(msg)["s"])
      WebSockets.send(ws, """<p data-hx-swap-oob='true' id='sr'> You picked $d </p>""")
    end
  end
end
```

You'll notice when you run this or similar code that the REPL blocks with a message that Julia is listening on the WebSocket. If you'd prefer to continue to work in the REPL while this program is running, perhaps to start further servers, the *HTTP* package provides another, non-blocking version of the function. It's the same, but using an exclamation mark; the call becomes:

```
WebSockets.listen!("127.0.0.1", PORT)
```

In the program you run on the server with the `systemd-run` command, you can start up any number of listening WebSocket servers with the non-blocking version of the call, but the final one should use the blocking version to keep the program alive and listening for connections.

The first argument to `WebSockets.listen()` is a handler function that takes a single `Websocket` argument; it's provided here as an anonymous function constructed with Julia's `do` keyword. This function is the one that does things with the incoming messages, so it should include the command that sends the result back to the client, which is accomplished by the `WebSockets.send()` call. The `for` loop keeps the connection open until the user closes it, which normally happens on closing the web page.

Listing 3 shows the typical pattern for using the HTTP package to communicate over WebSockets and is the pattern I use in the more complicated examples listed under HTMX Examples in Table 1.

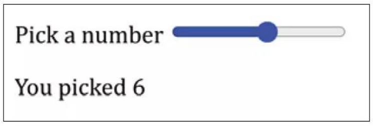
In more detail, the function `startserver()` in Listing 3:

- Begins to listen for WebSocket messages on the local machine coming through `PORT`
- Parses each message received, extracts the value associated with the name `s`, and assigns that to the variable `d`
- Immediately sends a message back to the client consisting of an HTML fragment with the value assigned to `d` interpolated

The example returned by the HTML fragment has a paragraph element with two attributes. The first, `hx-swap-oob` (with the optional `data` prefix to help keep the HTML standards compliant), tells HTMX that the fragment is to be inserted out of band and should replace the existing element with `id='sr'`. In the context of HTMX, "out of band" means that the element to be replaced is not the same as the element that sent the message: The slider itself is not to be replaced, but a different element – in this case, a paragraph.

Referring back to Listing 2, notice the existing empty paragraph with `id='sr'`. It's there as a placeholder, ready to be replaced after the user manipulates the slider. Figure 4 shows what this minimal example looks like in the browser after the user has placed the slider in a position corresponding to the number 6. The empty paragraph has been replaced by that constructed in Listing 3, containing the text *You picked 6*.

Listings 2 and 3 illustrate the mechanics of using the JavaScript HTMX library and the Julia HTTP package together to build an interactive page backed by Julia computations on a server. By using this method with a little HTML and JavaScript knowledge, you can create any application you can imagine. Often in these applications you will want to insert something other than text into the page: perhaps images, video, or sound. One obvious way to accomplish this is to have your Julia program save the media that it generates in a file on the server and send an element to the page that loads the file. This is the usual way that, for example, an image tag works, referring to the URL of the image to be loaded in its `src` attribute. This reasonable approach has at least two minor disadvantages: It requires an extra request to be sent by the client, which adds an interaction delay, and it litters your server with media


Pick a number

You picked 6

Figure 4: A simple client using HTMX.

files that you will need to clean up. If you delete them too soon, your users' pages might wind up trying to load resources that no longer exist, and if you leave them around too long, they will eat into the space on your server.

An elegant solution to these problems is to exploit an underutilized feature built in to HTML: the `data` URL **[21]**. With this feature you can send any kind of media a web browser can support (e.g., images, sounds, etc.) directly down the wire as part of the server's response, rather than sending the URL of a file that the browser subsequently has to request. No files need be created or stored. The media becomes part of the page, just as its text content is, rather than a separate resource.

To use this feature in your web applications, you'll need a way to encode the binary media data as text. Among the myriad ways to do this, the standard in browsers is Base64 encoding **[22]**, provided in Julia by the *IBase64* package.

Listing 4 shows the server program from **Listing 3** with a few lines added that create a plot from user input on the slider, encode it, and send it to the client for insertion on the page. The program imports the *Plots* and *Base64* packages to create and encode the graphics. At the beginning of `startserver()` an input-output buffer is assigned to the variable `io`. You'll need this buffer object as part of the conversion from a binary stream to a block of text.

Within the `for` loop you can see four new lines. First, the program creates a simple plot of the `sin()` function that incorporates the value selected by the user, assigning this plot to the variable `p`. The next two lines read the plot data into `io` as a PNG image, convert it to text (Base64-encoded), and store it in the variable `data`.

The final line in the `for` loop has an additional `send()` call that sends the plot to the client, replacing the existing image element (identified by its `id`) with an image element that embeds the plot in its `src` attribute. As you can see in the example, you have to tell the browser the mime type of the data and mention that it uses `base64` encoding.

Listing 4 also illustrates that you are not limited to sending one response for each message received but can send any number of them, targeting any set of elements on the page.

The client code for this example would be the same as in **Listing 2**, with the addition of the line

```
<img alt='' src='' id='plot'>
```

to create a placeholder element for the plot.

Figure 5 shows a screenshot of the browser after the user has manipulated the slider to select the number 8. If you implement the client and server from **Listings 3 and 4** and play with the slider in your browser, you'll see that the plot updates quickly, providing a good interactive experience, especially if you run the server on the same machine as your browser. Naturally, if the server and client are communicating over the Internet, the experience will be affected by the speed of the network, and the response will not always be smoothly interactive, which is an unavoidable drawback of all applications that depend on calculations carried out on a remote server. The principal way to minimize such issues is to avoid sending massive amounts of data in response to user actions. If you're sending an image, for example, take advantage of compressed file formats (e.g., JPEG) and limit the resolution to reduce the quantity of data as much as possible.

The other source of delays in interactivity is the time needed to carry out the calculations on the server. In this case, Julia's speed helps and is one reason it's a good choice for these applications: For the typical calculations supporting the types of pedagogical purposes that my examples serve, the calculations are sub-second, and calculation time is not a major factor limiting the experience of interactivity.

You can implement the example in this section with the PlutoSliderServer method, as well; which one you might choose depends on your plans for further development, whether you enjoy working with Pluto notebooks, and whether having an HTML page that can pass validation is important to you. For an example of an application that uses the techniques described in this section but that would be cumbersome to build with PlutoSliderServer, see the vortex dynamics example in **Table 1**. You can examine the source of the client in the usual way, and the page explains how to obtain the source code for the server component.

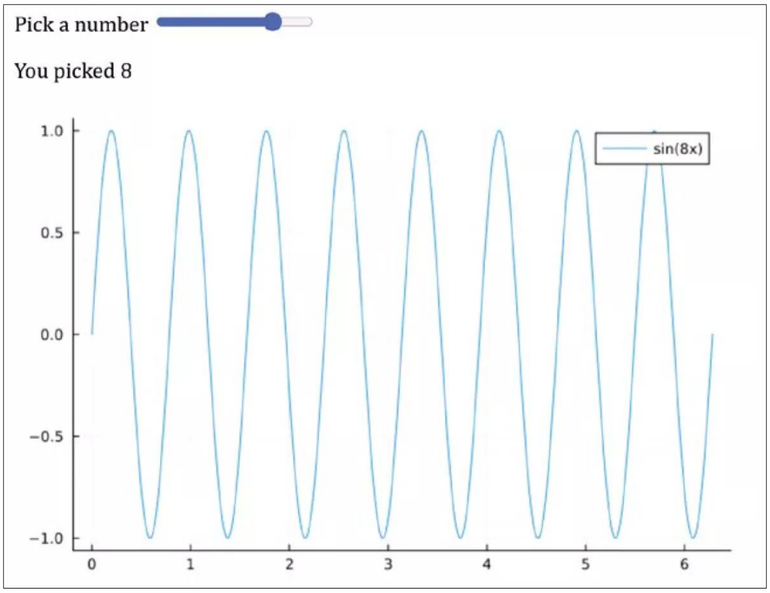
Listing 4: Server with Graphics

```
using HTTP, JSON, Plots, Base64
const PORT = 8660

function startserver()
  io = IOBuffer();
  x =  0.0:2π/1000:2π
  WebSockets.listen("127.0.0.1", PORT) do ws
    for msg in ws
      d = Meta.parse(JSON.parse(msg)["s"])
      @info d
      @info typeof(d)
      WebSockets.send(ws,  """<p hx-swap-oob='true' id='sr'>You picked $d</p>""")
      p = plot(x, sin.(d .* x); label="sin($(d)x)")
      show(io, MIME"image/png"(), p)
      data = base64encode(take!(io))
      WebSockets.send(ws,  """<img data-hx-swap-oob='true' id='plot' src='data:image/png;base64,$data' alt='sin(1/x)'>""")
    end
  end
end
```

A Final Example

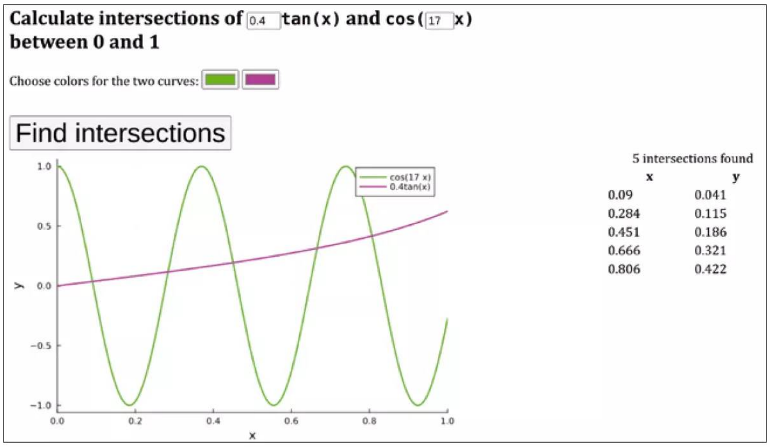
One more example, expanding a bit on the previous one, will serve to demonstrate a few more controls and show how to use styles to lay out the page. Additionally, this example actually does something almost useful. The mathematical problem of finding the intersections of trigonometric functions typically has only numerical solutions. **Figure 6** is a browser screenshot showing an application that lets the user pick two parameters: one for the amplitude of the tangent function and the other for the frequency of a cosine function. The user can also select two colors with two color pickers. On clicking the *Find intersections* button, two new elements appear on the page: a graph of the two functions in the selected colors and a table listing all of their intersections within the range (0, 1). **Listing 5** contains the complete HTML code for the client page, aside from the header and other HTML administrivia, and **Listing 6** has the complete server program.



Pick a number

You picked 8

Figure 5: An HTMX client with an image.



Calculate intersections of 0.4 tan(x) and cos(17 x)
between 0 and 1

Choose colors for the two curves: 🟩 🟪

5 intersections found

x	y
0.09	0.041
0.284	0.115
0.451	0.186
0.666	0.321
0.806	0.422

Figure 6: An HTMX client solving a mathematical problem.

By this point, the function of most of what's in these programs should be clear. The server code contains nothing fundamentally new: just some additional variables and the use of the *Roots* package **[23]** to calculate the function intersections. Note the inclusion of style attributes in the returned HTML fragments and the incorporation of variables in plot labels and the table caption. The first two lines in the outer `for` loop extract the two variables set by the color selectors, leaving them as strings for direct insertion in the `plot()` call.

Listing 5 illustrates the use of forms with HTMX. Rather than placing the HTMX communication attributes in an input element as you saw in **Listing 2**, you can place them in an opening `form` tag. When the user hits the form submission button, all the variables defined by input controls within the form are sent over the WebSocket in a single message.

Listing 5: Client for Figure 6

```
<form data-hx-ext="ws" data-ws-send data-ws-connect="ws://127.0.0.1:8660">
  <h2>Calculate intersections of <code><input style='width:3em;' type="number" name="t" min="0.1" max="1" value='1' step='0.1'>tan(x)</code> and
     <code>cos(<input style='width:2.5em;' type="number" name="c" min="1" max="40" value='4'>x)</code><br> between 0 and 1</h2>
  Choose colors for the two curves:
  <input type='color' name='c1' value='#990000'>
  <input type='color' name='c2' value='#000099'><br>
  <input type='submit' style='font-size:2em; margin-top:1em;' value='Find intersections'><br>
  <img alt='' src='' id='plot'>
  <table id='ztable'></table>
</form>
```

Go Forth and Code

The virtues that have made Julia so successful in the realm of scientific computing and engineering – ease of development combined with uncompromised performance – also make it a good choice for the back end of web applications. In this article, you've learned about two approaches to making interactive websites for education, information, and entertainment. You might decide to adopt one or the other (or neither!) or, like me, apply the two approaches to different projects. I've found working with Julia in concert with web technologies to be one of my most enjoyable and rewarding adventures in programming. I hope you will have as much fun as I did with my projects in building things that I can't imagine. Please share news of your creations by dropping me a line through email or in comments on my website. ■

Listing 6: Server for Figure 6

```
using HTTP, JSON, Base64, Roots
const PORT = 8660

function startserver()
  io = IOBuffer();
  x =  0.0:0.001:1.0
  WebSockets.listen("127.0.0.1", PORT) do ws
    for msg in ws
      lc1 = JSON.parse(msg)["c1"]
      lc2 = JSON.parse(msg)["c2"]
      c = Meta.parse(JSON.parse(msg)["c"])
      t = Meta.parse(JSON.parse(msg)["t"])
      f1(x) = cos(c * x)
      f2(x) = t * tan(x)
      f(x) = f1(x) - f2(x)
      z = round.(find_zeros(f, (0, 1)), digits=3)
      y = round.(f1.(z), digits=3)
      ztable = []
      for zero in zip(z, y)
        append!(ztable, "<tr><td>$(zero[1])</td><td>$(zero[2])</td></tr>")
      end
      ztable = join(ztable)
      p = plot([f1 f2]; xrange=(0, 1), lc=[lc1 lc2], lw=2, xlabel="x", ylabel="y", label=["cos($c
         x)" "$(t)tan(x)"])
      show(io, MIME"image/png"(), p)
      data = base64encode(take!(io))
      WebSockets.send(ws,  """<img data-hx-swap-oob='true' id='plot' src='data:image/
         png;base64,$data' alt='sin(1/x)'>""")
      WebSockets.send(ws, """<table data-hx-swap-oob='true' id='ztable' style='width:13em;
         float:right;'> <caption> $(length(z)) intersections found </caption> <tr><th>x</th><th>y</
         th></tr> $ztable </table>""")
    end
  end
end
```

Info

[1] Julia language: [https://julialang.org/]

[2] "Fast as Fortran, Easy as Python" by Lee Phillips, *ADMIN*, issue 50, 2019, [https://www.admin-magazine.com/Archive/2019/50/Julia-Fast-as-Fortran-easy-as-Python]

[3] "New ways to compile Julia" by Jeff Bezanson, *JuliaCon 2024*: [https://info.juliahub.com/blog/new-ways-to-compile-julia-blog]

[4] Phillips, Lee. Scientific computing's future: Can any coding language top a 1950s behemoth? *Ars Technica*, May 2014: [http://arstechnica.com/science/2014/05/scientific-computings-future-can-any-coding-language-top-a-1950s-behemoth/]

[5] Julia joins petaflop club, *HPCWire*, 2017, [https://www.hpcwire.com/off-the-wire/julia-joins-petaflop-club/]

[6] "Rankings of Julia Books on Amazon," by Lee Phillips, 2024: [https://lee-phillips.org/amazonJuliaBookRanks]

[7] Phillips, Lee. *Practical Julia.* No Starch Press, 2023: [https://nostarch.com/practical-julia]

[8] PlutoSliderServer.jl: [https://github.com/JuliaPluto/PlutoSliderServer.jl]

[9] Pluto.jl: [https://github.com/fonsp/Pluto.jl]

[10] HTMX – High power tools for HTML: [https://htmx.org/]

[11] Jupyter: [https://jupyter.org/]

[12] "An Introduction to Pluto" by Lee Phillips, *LWN.net*, November 2020: [https://lwn.net/Articles/835930/]

[13] PlutoUI.jl: [https://github.com/fonsp/PlutoUI.jl]

[14] Pkg: [https://docs.julialang.org/en/v1/stdlib/Pkg]

[15] PlutoUI widgets: [https://featured.plutojl.org/basic/plutoui.jl]

[16] HypertextLiteral overview: [https://juliapluto.github.io/HypertextLiteral.jl/stable]

[17] systemd-nspawn: [https://wiki.archlinux.org/index.php/Systemd-nspawn]

[18] Javis: [https://github.com/Wikunia/Javis.jl]

[19] HTTP.jl: [https://github.com/JuliaWeb/HTTP.jl]

[20] HTMX WebSocket extension: [https://htmx.org/extensions/ws]

[21] Data URLs: [https://developer.mozilla.org/en-US/docs/Web/URI/Schemes/data]

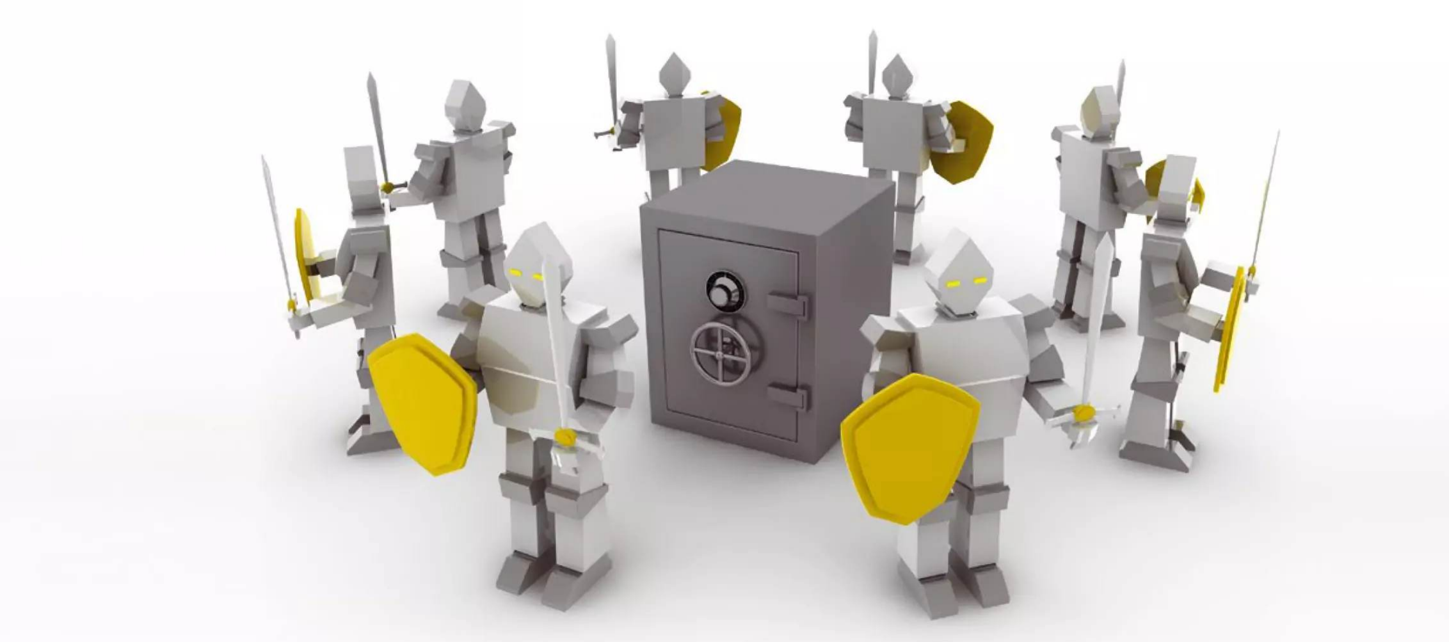
[22] Base64: [https://developer.mozilla.org/en-US/docs/Glossary/Base64]

[23] Roots.jl: [https://github.com/JuliaMath/Roots.jl]

[24] Phillips, Lee. *Einstein's Tutor*. PublicAffairs, 2024: [https://www.amazon.com/gp/product/1541702956?tag=hacboogrosit-20]

Author

Dr. Lee Phillips is a theoretical physicist and writer who has worked on projects for the Navy, NASA, and the DOE on laser fusion, fluid flow, plasma physics, and scientific computation. He has written numerous popular science and computing articles and research papers. His most recent books are *Practical Julia* [7] and *Einstein's Tutor* [24].



Lead Image © tarokichi, 123RF.com

ASM tools and strategies for threat management

Choose Your Armor

The tools used in attack surface management help identify attack surfaces more precisely and respond to changes in risk situations. By Wilhelm Greiner

Identifying and exploiting vulnerabilities has been part of the attackers' trade for thousands of years. Just think of Hagen von Tronje, for example, who snuck up behind Siegfried, took aim with his spear, and murdered the purportedly invulnerable hero. He knew about the weak point in Siegfried's protective armor. Literary historians talk about Siegfried's death as an intrigue or honor killing, but as security professionals know, it was yet another case of inadequate attack surface management. In this article, I shed light on what is important in terms of IT security when reducing the attack surface.

ASM

Security analysts and providers use the term “attack surface management” (ASM) to describe tools and software-as-a-service (SaaS) offerings that are intended to enable enterprises – large corporations in particular – to identify their attack surfaces more precisely and respond more quickly to changes in their risk situation (see the “Prevention and NIS2” box). In their Leadership Compass publications on ASM [1], analysts at KuppingerCole states that ASM has “emerged as a crucial discipline that enables proactive cybersecurity strategies, mitigating risks by reducing an organization's exposure to potential attacks.”

KuppingerCole defines the attack surface as the “totality of all possible entry points within an organization, as well as the digital infrastructure of its subsidiaries and partners.” In addition to hardware, software, storage devices, and networks on-site and in the cloud, the analysts also include identities (of users, accounts, and devices) that an attacker could exploit to block services, gain unauthorized access, carry out attacks, or compromise sensitive data. The problem is that the attack surface is constantly changing, if only because the components of an IT environment are constantly changing. According to KuppingerCole, due diligence therefore requires these attack surfaces to be monitored and evaluated 24/7. Similarly, analysts at Forrester Research in 2022 described ASM as “the process of continuously discovering, identifying, inventorying, and assessing the exposures of an entity's IT asset estate” [2] and reported on success stories with ASM users praising the superior overview, the time savings, and the ability to prioritize risks.

Two ASM Variants

Market observers such as Gartner, Forrester, and KuppingerCole distinguish between two types of ASM: external (EASM) and cyber asset (CAASM). CAASM is designed to empower organizations to record all internal and external assets, primarily through API integration with existing

Prevention and NIS2

Dennis-Kenji Kipker, Professor of IT Security Law at the Bremen University of Applied Sciences, points out that NIS2 [the successor to the 2016 European Network and Information Systems Directive] is based on the principle of prevention; therefore, you look at what the attack vectors are so that if you do get compromised, you have an emergency management system in place, which ultimately also includes all risk management measures. From this point of view, he says, ASM can be interesting for some companies, although it is not a legal requirement.

tools such as inventory. The data collected and consolidated by CAASM is used to determine the type and extent of vulnerabilities, so they can be analyzed and continuously monitored. Gartner [3] includes Armis, Axonius, Balbix, JupiterOne, Lansweeper, OctoXLabs, runZero, and ThreatAware in its list of CAASM providers.

In times of ransomware and other cyberattacks, resources exposed to the Internet are particularly at risk, and IT teams nowadays sometimes only have hours instead of days to patch vulnerabilities. The second ASM variant, EASM, focuses on this particularly explosive defense case.

According to Gartner, the scope of EASM includes misconfigured public cloud services and servers, as well as exposed company data such as login credentials and vulnerabilities in third-party software code that an attacker could exploit – think of the SolarWinds compromise in 2020 or the recent attack on the open source XZ Utils toolbox, which Andres Freund discovered by chance.

The EASM market segment is home to various major players such as IBM (with the acquisition of Randori in 2022), Google Cloud (with the acquisition of Mandiant, also in 2022), and Microsoft. They are likely to be more concerned about getting their own attack surfaces under control. These major players are joined by numerous security specialists, including CrowdStrike, Group-IB, Palo Alto Networks, and Trend Micro.

KuppingerCole assumes that the two types of ASM will sooner or later merge into a holistic ASM. In the meanwhile, user companies will certainly not want to maintain and pay for separate tools, processes, or cloud services to monitor both ends of the same asset.

X Marks the Spot

King Gunther and his henchman Hagen trick Siegfried's wife Kriemhild into believing that war with the Saxons is imminent. In this way, Hagen is able to coax Siegfried's secret out of her: Under the pretext that he can then protect Siegfried all the better, he persuades Kriemhild to sew a cross onto Siegfried's robe at the critical point. Social engineering has always been one of the most effective weapons in the attacker's arsenal. Villains today have many different ways to dupe a victim, and loopholes can often be detected easily from a distance, because many companies expose vulnerabilities online with almost Kriemhild-like recklessness.

How does ASM prevent malicious actors from finding so many digital crosses that they can shamelessly exploit? First, you need to distinguish ASM from legacy vulnerability management, which relies on network-wide scans of the in-house IT inventory to compare the risk assessment results with the Common Vulnerability Scoring System (CVSS). Compared with the traditional approach, ASM tools and services enable faster and, above all, more precise prioritization. According to KuppingerCole, ASM allows organizations to take a risk-based approach and strategically focus their resources on the areas that are potentially most at risk.

This promise sounds pretty familiar. One suspicion involuntarily rears its head like an annoyed dragon. Could it be that resourceful marketeers have poured old vulnerability management wine into new ASM wineskins? Stefan Strobel [4], head of the security consultancy cirosec, pointed out that the basis of ASM is a vulnerability scanner (i.e., established technology). According to Strobel, though, ASM also covers companies with a decentralized organization that increasingly rely on cloud services and quickly lose track of their IP addresses in the process. You must first find out what you need to scan before scanning by asking questions such as: Which domains are in use? Who registered them? With which provider? On what subnet?

What might sound simple, at first, can quickly prove to be treacherous ground in large IT environments. By way of an example, when Forrester surveyed ASM users for its report, the security expert at a used car marketplace said that the ASM tool had found 50 percent more assets than they thought they had. Whereas dragonslayers of old only had to worry about a single weak spot, protecting attack surfaces in companies today is initially like looking for a needle in a haystack.

Cloud Computing Aggravates Risk

Security experts consider cloud computing an important driver for the need for digital searches. Common practice in some companies is for specialist departments to subscribe to public cloud services on demand, often without informing the in-house IT department. Shadow IT of this kind exacerbates the situation for the IT team, because it becomes difficult to maintain an overview of the IT inventory and clutters access paths, pushing tried and trusted routines to their limits.

If the IT team has previously carried out vulnerability scans on a weekly basis, they could potentially miss a large part of this cloud dynamic. ASM, on the other hand, aims to scan the entire Internet to understand the elements of a company's infrastructure. For example, an ASM tool can map certificates of a cloud service to a company, even if the IT manager or chief information security officer (CISO) is not aware a department has booked a cloud service.

The obvious reaction to a cluttered view is to just use Shodan, but ASM providers warn against this method, pointing out that it is almost impossible to consider all threats that could harm a company or its market presence. ASM specialists point to their expertise in detecting actively used exploits and often maintain research teams that investigate how threat actors carry out attacks or even infiltrate the dark web scene.

Dealing with Identified Risks

Security expert Richard Werner from Trend Micro distinguishes two types of contingency:

- A risk has been discovered and a decision must be made about how to deal with it.
- The risk has materialized (i.e., an attack is in progress).

Werner emphasizes that according to the NIS2 regulation, corporations need to prepare for both scenarios, because it simply cannot be assumed that all risks have been ruled out.

For a long time, security teams used to think of IT security as something like a medieval fortress, with the good guys on the inside, the bad guys on the outside, and the castle wall and moat in between. However, this fortress-style isolation has long since proven to be a deceptive myth – not to mention that the best-known stories suggest that completely eliminating vulnerability is a myth.

According to a study by Trend Micro, an average of 11 percent of assets in companies are highly vulnerable. A security team needs to monitor these assets particularly closely to be able to take action quickly if signs of risk levels rise. The Japanese security provider refers to this approach as "attack surface risk management," which is pretty much the same as ASM.

Functional Scope of ASM

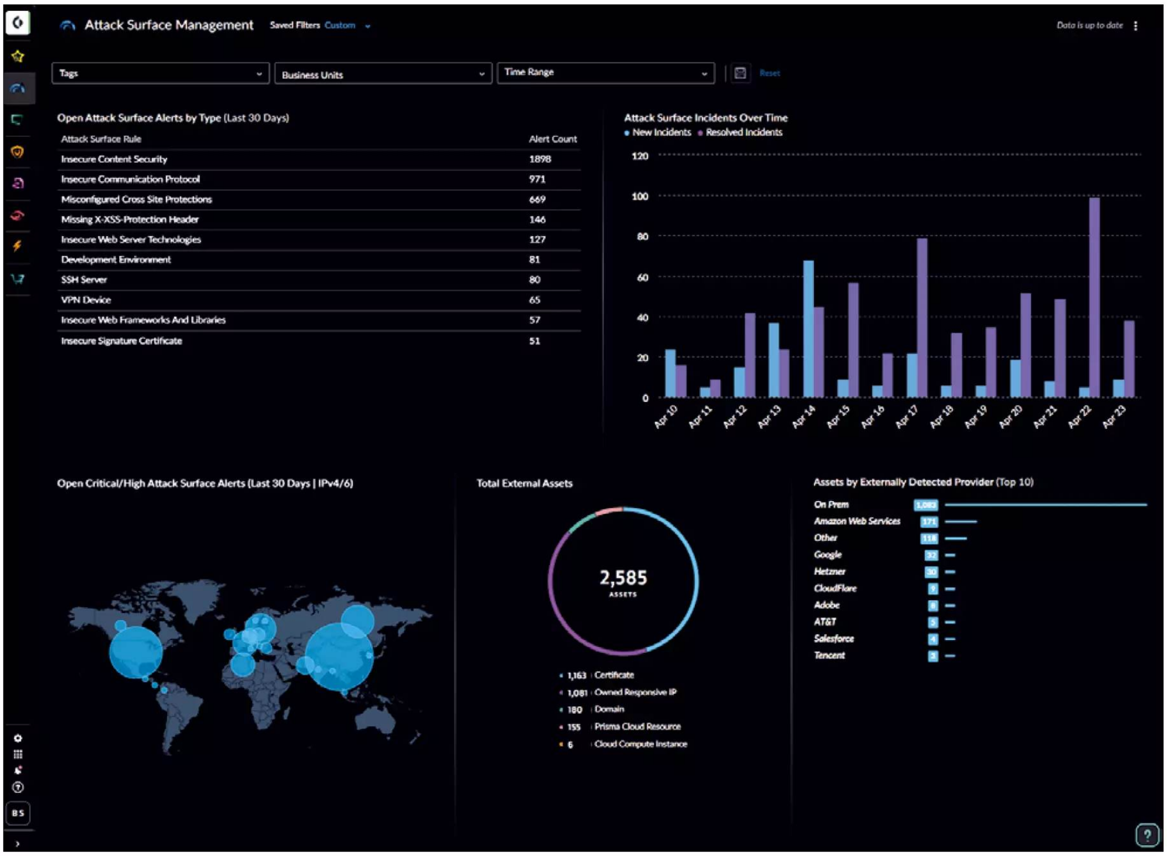
According to KuppingerCole, ASM comprises five areas of responsibility:

- Detecting vulnerable assets
- Creating profiles and evaluating vulnerabilities
- Prioritizing checkpoints on the basis of risk assessment
- Monitoring vulnerabilities on an ongoing basis
- Fixing vulnerabilities – or at least help fix them

The scopes of these five tasks lead to a number of key and auxiliary functions aimed at analyzing, visualizing, monitoring, and, if possible, automatically reducing the attack surface (see the "ASM Functions" box).

How can you separate the wheat from the chaff in the large numbers of ASM offerings around today? According to Strobel, it is a question of what belongs to the corporate network and what does not. Cirosec tested several ASM solutions, some of which performed extremely poorly according to Strobel. One tool did not even correctly map a wildcard for web addresses. As to the number of vulnerabilities a tool finds, established vulnerability engines (e.g., Qualys, Rapid7, and Tenable) are readily available on the market. According to Strobel, however, other tools cover aspects that are less exciting (e.g., websites without an imprint or cookie banners) but can fail to find the genuinely relevant risks.

The core component is always an approach to automated validation of security vulnerabilities that must put


Attack Surface Management
Saved Filters Custom
Data is up to date
Tags
Business Units
Time Range
Reset
Attack Surface Incidents Over Time
New Incidents
Resolved Incidents
120
100
80
60
40
20
0
Apr 10
Apr 11
Apr 12
Apr 13
Apr 14
Apr 15
Apr 16
Apr 17
Apr 18
Apr 19
Apr 20
Apr 21
Apr 22
Apr 23
Open Critical/High Attack Surface Alerts (Last 30 Days | IPv4/6)
Total External Assets
2,585
ASSETS
1,163 Certificate
1,081 Owned Responsive IP
180 Domain
155 Prisma Cloud Resource
6 Cloud Compute Instance
Assets by Externally Detected Provider (Top 10)
On Prem 1,082
Amazon Web Services 171
Other 118
Google 32
Hetzner 30
CloudFlare 9
Adobe 8
AT&T 5
Salesforce 4
Tencent 3
BS
?

Open Attack Surface Alerts by Type (Last 30 Days)

Attack Surface Rule	Alert Count
Insecure Content Security	1898
Insecure Communication Protocol	971
Misconfigured Cross Site Protections	669
Missing X-XSS-Protection Header	146
Insecure Web Server Technologies	127
Development Environment	81
SSH Server	80
VPN Device	65
Insecure Web Frameworks And Libraries	57
Insecure Signature Certificate	51

Figure 1: Attack surface management platforms provide security teams with an overview of the threat situation, including external (e.g., cloud) assets, from a dashboard.

ASM Functions

Key ASM functions include:

- Reduction of the attack surface: zero-trust architecture, strong authentication, granular authorization, network segmentation, and interoperability with security tools
- Visualization of the attack surface: dashboards (Figure 1), mapping to MITRE ATT&CK
- Integration with cyberthreat intelligence (CTI) feeds: aggregation of data from different sources to better understand the attack surface
- Attack surface analysis: aggregation and evaluation of extensive databases, if necessary, with machine learning
- Automated attack surface management: continuous detection, analysis, and monitoring of assets and their vulnerabilities

Supplementary or emerging functions include:

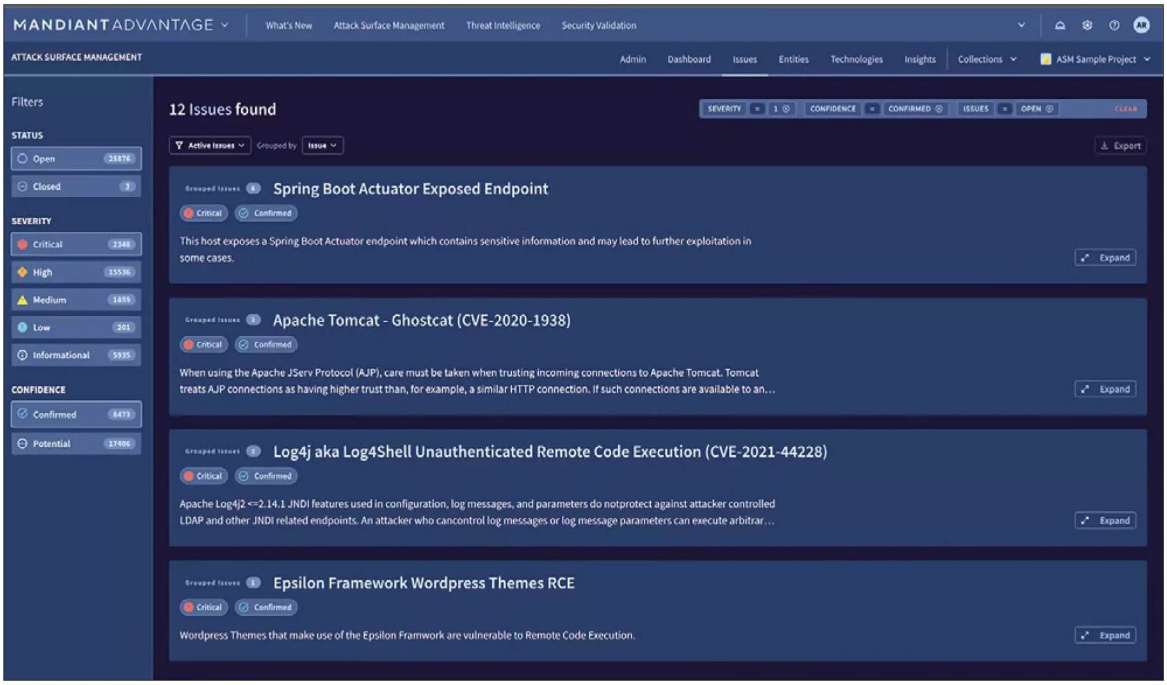
- Cloud: monitor software- and infrastructure-as-a-service (SaaS and IaaS) resources; according to KuppingerCole, state-of-the-art ASM systems are themselves hosted in the cloud
- IoT monitoring, which also includes industrial IoT
- Dark web monitoring: Some ASM vendors have infiltrated dark web forums and attacker groups such as ransomware-as-a-service sites
- Brand protection: monitoring for brand misuse, from DNS manipulation to monitoring social media channels
- DevSecOps integration: ASM should be integrated into DevOps processes at an early stage
- Threat modeling and attack simulation: the use of ASM information to simulate attacks
- Compliance: comparison with legal requirements such as the General Data Protection Regulation (GDPR) or, if applicable, NIS2

IT or security teams in a position to focus on those attack surfaces that a professional attacker would tend to exploit. IBM refers to a Forrester survey, according to which users of the IBM Randori tool require 90 percent less time for vulnerability scans averaged over the year.

ASM specialists such as Google cloud subsidiary Mandiant recommend a comprehensive platform that monitors the dark web, vulnerabilities, and threat data and offers functions to identify and prioritize risks (Figure 2). According to the US provider, up-to-date threat intelligence in particular plays an important role here. Google Cloud's ASM platform combines external resources with Mandiant's front-line threat data to identify the current behavior of relevant threat actors or potential malware continuously.

According to the provider, the prioritization logic considers numerous aspects: shadow IT or unknown systems with their vulnerabilities, misconfigurations of cloud resources, unauthorized developer repositories, suspicious issues such as typosquatting (misuse of typos or word similarities in web addresses), Punycode domains (use of non-ASCII characters in domain names for attack purposes), brand abuse in mobile device apps, web application endpoints, and the security situation of third-party providers.

According to security specialists Group-IB, ASM users set great store by scanning user data; the inclusion of dark web, malware, and leak information in overviews; API and user-defined notification options; and detailed remediation instructions. According to Group-IB, the in-house ASM product follows the user's


MANDIANT ADVANTAGE
What's New
Attack Surface Management
Threat Intelligence
Security Validation
AR
ATTACK SURFACE MANAGEMENT
Admin
Dashboard
Issues
Entities
Technologies
Insights
Collections
ASM Sample Project
Filters
STATUS
Open 25876
Closed 3
SEVERITY
Critical 2348
High 15536
Medium 1855
Low 201
Informational 5935
CONFIDENCE
Confirmed 8473
Potential 17406
SEVERITY = 1
CONFIDENCE = CONFIRMED
ISSUES = OPEN
CLEAR
Active Issues
Grouped by
Issue
Export
Grouped Issues
Critical
Confirmed
Expand

12 Issues found

Spring Boot Actuator Exposed Endpoint

This host exposes a Spring Boot Actuator endpoint which contains sensitive information and may lead to further exploitation in some cases.

Apache Tomcat - Ghostcat (CVE-2020-1938)

When using the Apache JServ Protocol (AJP), care must be taken when trusting incoming connections to Apache Tomcat. Tomcat treats AJP connections as having higher trust than, for example, a similar HTTP connection. If such connections are available to an…

Log4j aka Log4Shell Unauthenticated Remote Code Execution (CVE-2021-44228)

Apache Log4j2 <=2.14.1 JNDI features used in configuration, log messages, and parameters do notprotect against attacker controlled LDAP and other JNDI related endpoints. An attacker who cancontrol log messages or log message parameters can execute arbitrar…

Epsilon Framework Wordpress Themes RCE

Wordpress Themes that make use of the Epsilon Framwork are vulnerable to Remote Code Execution.

Figure 2: The evaluation of attack surfaces in the context of individual IT environments is a key function of any ASM system. © Google Cloud/Mandiant [5]

interest patterns and focuses more or less on specific resources. The idea is to allow the system to assess risks in a dynamic way; that is, determine the relevance of a resource or its threat level for the company on the basis of variable statistical values.

Mandiant emphasizes the value of integration with the user infrastructure to improve transparency, which makes it easier to set priorities and tackle critical security problems effectively.

ASM in Industrial Use

As is so often the case, industrial networks pose particular difficulties for the security team; industrial companies are exposed to dangers in two ways, because their potential attack surface includes both information and operating technologies (IT and OT). Increasing automation and networking of machines and systems further increases this attack surface. As OT security specialists such as Kaspersky or Claroty also point out, industrial environments come with one hugely critical flaw: Legacy devices, such as plant control systems, often run on operating systems that are so badly outdated they no longer receive any updates or security patches. These challenges make protecting industrial environments a duel with a fire-breathing dragon.

According to providers such as CrowdStrike, the very first campaign in industrial dragon control has to be a comprehensive inventory of the industrial environment. CrowdStrike refers to this as extended Internet of things (XIoT). The inventory is intended to provide clarity with tests and queries by penetrating deep into industrial control system (ICS) subnets and identifying and fingerprinting all XIoT devices regardless of their protocols. To do so, it determines device information such as type, manufacturer, location, location name, class, level in the Purdue reference model, protocol, operating system, and so on. According to CrowdStrike, this scan is the only way to ensure essential in-depth insights into the attack surface and potential damage in the event of an attack.

That said, many experts warn that, although it is essential to inventory the IT and OT inventory, it will often not be possible to register all assets fully. The logical consequence is that an industrial company's security team needs to build skills to keep up with cybercriminal weapons. Ultimately, it is about being able to find out just as quickly as the attackers in the IT and OT areas where the company has vulnerabilities and what they are – and to react quickly.

This reaction is proving extremely difficult in the industry, and not just because of the stubborn old devices mentioned. Patching current systems is also a major challenge, except in plain vanilla IT environments. Security in the OT context traditionally means operational reliability (i.e., the system running with as few interruptions as possible), which is why OT experts emphasize the key role of risk-based vulnerability management. According to Claroty, in industry, the goal is not about shutting down every known vulnerability, but rather about checking how likely it is that attackers will exploit the vulnerabilities in your systems. From these findings, targeted remedial action or, if this is not possible, at least establishing control mechanisms to mitigate the risk is essential.

According to Cirosec boss Strobel, however, ASM is only of limited use in an industrial context, because ASM generally only looks at externally accessible components. In OT environments, however, often nothing can be seen from the outside. Digital twins, cloud connections, interfaces, or remote maintenance access are increasingly being set up in the scope of Industry 4.0, and an ASM system would, one hopes, find them. According to Strobel, however, these problems are peripheral, saying the core problem usually lies in the company's internal OT or IT network.

To minimize attack surfaces in the industrial environment, security experts always give the same advice: Network segmentation, ideally along business processes, is the ideal solution and the only alternative if a system can no longer be patched. According to the experts, industrial companies need to supplement this segmentation with protective measures wherever possible, including intrusion detection and prevention systems, firewalls, and antivirus software for OT environments. No less important is regular – or preferably continuous – monitoring and analysis of network activities.

Finally, experts repeatedly point to those attack surfaces that stem from human vulnerability. In the industrial sector in particular, for example, operating personnel like to share passwords for a plant control system within the team – a nightmare for any security officer. An industrial company that consistently implements the basics of cyber hygiene can significantly reduce the number of vulnerable points.

AI Support for ASM

As is the case almost everywhere in the security industry, various manufacturers now laud the benefits of AI when it comes to ASM, which does not usually refer to large language models (LLMs) and the currently hotly debated generative AI, but rather to self-learning statistical analyses (machine learning, ML).

CrowdStrike, which has been working with AI since 2011, introduced in 2021 ExPRT.AI, an AI-supported assessment system to better prioritize threat defense. The ExPRT.AI rating is dynamically adjusted on the basis of the current exploit status and threat data.

In good news, according to experts such as Sergej Epp from Palo Alto Networks, ML support works much better in industrial networks than in IT because OT has relatively static communication patterns. If an ML-supported security system monitors the OT network over a training period of two or three days, says Epp, it knows which control device belongs to which protocol and which system. The tool can then automatically recommend to the firewall that only one protocol should be allowed in this segment to minimize the scope of an attack.

According to security experts, LLMs can also be beneficial for remedial measures, because they can summarize large volumes of information from a wide variety of sources. This ability of language models to provide comprehensive context-related advice could, for example, help speed up remediation tasks in the future.

The Limits of ASM

The KuppingerCole report stated that, as a discipline, ASM takes the attackers' point of view. In fact, some vendors describe their ASM services as continuous red teaming. However, according to cirosec boss Strobel, these statements are not entirely accurate and points out that you have to put it into perspective because an advanced persistent threat (APT) group also takes a close look at a company's employees or uses spear phishing, which is outside the scope of ASM.

In particular, Strobel vehemently disagrees with the claim that ASM allows quasi-automated red teaming, which he says is something completely different. Rather, it is a project that takes months of work, including spear phishing, social engineering, physical hardware that is connected to the power socket in the company, proprietary malware for the backdoor, and so on. He emphasized that this process cannot be automated and that anyone who makes this argument is misusing the term "red teaming." Additionally, red teaming is about testing the blue team's (i.e., the defenders') ability to react, which is a completely different goal from ASM. Despite such marketing exaggerations, Strobel believes that ASM can still be useful, especially for larger companies with subsidiaries and foreign locations that lack an overview of their IT environment.

Conclusions

Keeping an eye on all areas of attack. The Nibelungs found this out to their own downfall. You can take some warnings from the Song of the Nibelungs that still apply today. The most dangerous attackers are people like Hagen von Tronje, who proceed with patience and thorough planning; today, this approach would be referred to as APT. The attacker is even more dangerous if they have detailed knowledge of attack surfaces – not least through carelessly disclosed internal information. Finally, the murder of Siegfried shows that exploiting even a small security vulnerability can have fatal consequences if the attacker exhibits determination and purpose.

Software solutions and SaaS offerings for attack surface management can support the defense team in their task. According to experts, ASM tools are particularly effective in large, confusing, and constantly changing IT landscapes and are helpful when you can no longer see the trees for the leaves, because they make it easier for security teams to gain an overview and set priorities promptly to close the most critical gaps as quickly as possible.

ASM can also play a role in industrial environments – at least if the industrial network is open to the Internet in the scope of Industry 4.0 or smart factory initiatives. Although cyber risk management is required by law (e.g., in NIS2), ASM is not. However, any company that becomes aware of attack surfaces as part of its risk management must respond. After all, you do not want potential attackers to find an attractively large target. ■

Info

[1] "Attack Surface Management" by O. Celik and J. Tolbert, KuppingerCole Analysts AG, September 2023, [https://www.kuppingercole.com/research/lc81218/attack-surface-management]

[2] "Forrester's New Research On Attack Surface Management" by Jess Burn, Forrester Research Inc., January 2022, [https://www.forrester.com/blogs/announcing-forresters-new-research-on-attack-surface-management-asm/]

[3] External attack surface management, Gartner Peer Insights: [https://www.gartner.com/reviews/market/external-attack-surface-management]

[4] Stefan Strobel: [https://cirosec.de/en/news/author/stefanstrobel/]

[5] Google threat intelligence: [https://cloud.google.com/security/products/threat-intelligence]

Purdue Model for industrial networking

Safety Dance

The Purdue Model maps the challenges of networking industrial systems to five levels, helping to target and mitigate risk and address vulnerabilities. We look at the Purdue Model in detail, investigate an implementation tool, and explain the role of zero trust. By Gerd Pflüger

By way of an example of how digital transformation has fundamentally changed the way things are done, just consider how buildings are managed and operated. This development is particularly advanced in the area of facility management from the use of what is known as the "digital ceiling." This technology integrates different building systems such as lighting, heating, cooling, and security on a single, intelligent network platform. The benefits offered by this kind of integration are huge: Besides significant efficiency boosts attributable to the automation of building services, the technology also improves cost efficiency in facility management.

In this article, I focus on the challenges of the digital ceiling and of establishing the framework for a more in-depth discussion of the Purdue Model and its application in the context of modern facility networking. I then go on to explore how the Purdue Model helps ensure secure and efficient networking in facility management by highlighting specific security strategies for each level of the technologic implementation.

Digital Ceiling

The digital ceiling relies on IP-based communication and is powered by Power over Ethernet (PoE), enabling direct connections over the facility network with state-of-the-art Ethernet switches. Although these technologic advances open up new options, they also harbor specific risks. Centralized control of a zoo of systems on a network extends the attack surface for cyberattacks, with a greater risk of data leaks or even manipulation of critical systems.

Many security risks (e.g., misconfigured access controls and poor authentication mechanisms) lead to a greater risk of unauthorized access. Moreover, widespread data acquisition and processing by smart building systems poses a considerable challenge in terms of data protection, particularly with regard to personal data.

Facility Networking

The differentiation between information technology (IT) and operational technology (OT) plays a central role in understanding modern facility networking systems or industrial networking in general. IT systems are primarily responsible for managing and processing data and information, and IT infrastructures typically consist of servers, computers, software and databases that are mainly geared towards supporting business processes. In contrast to this, OT focuses on directly controlling and managing physical devices. These devices, including industrial control systems, sensors, and robots, are generally used in

Lead Image © James Thew, Fotolia.com

production facilities and service provision and help to automate physical processes.

The advanced implementation of OT in facility management, as exemplified by the digital ceiling, impressively demonstrates the interaction of these technologies. The digital ceiling, an ecosystem of interconnected devices and systems, leverages the benefits of both IT and OT to create an efficient, smart environment. Modern LED luminaires controlled by PoE, adaptive climate controls with sensors for temperature and air quality, and integrated security systems with surveillance cameras and access controls are just a few examples of the harmonization of IT and OT on a unified network.

These systems are not only energy-efficient, but also dynamically adapt their functions to the ambient conditions and the presence of people, all of which helps to optimize power consumption and improve user comfort. Of course, this convergence brings with it specific security problems that simply do not exist in traditional IT environments:

- Patching and updates: Many OT devices are designed for longevity and continuous operation and are less likely to see regular software updates, which can aggravate risk by the use of outdated software with known vulnerabilities.
- Agent-based security measures: Because of the limited computing capacities of many OT devices, the use of conventional security software is typically impossible. Instead, you need customized products that were specially developed for use in OT environments.
- Over-the-air (OTA) updates: Although OTA programming provides a convenient way to update software, it needs a secure design to prevent manipulation en route.
- Proprietary systems: Many OT systems are based on proprietary technology, which makes it difficult for external security experts to check and secure them. Additionally, vendor tie-in can lead to challenges if support is discontinued.
- Physical security risks: OT devices are often exposed to physical threats, which can range from environmental influences to direct tampering.

Effectively securing networked OT systems requires a differentiated approach that takes into account both technical and operational specifics. You will need targeted security measures to address the challenges of OT security in modern building environments while maximizing the benefits of IT/OT convergence.

Purdue Model

The Purdue Model, formally known as Purdue Enterprise Reference Architecture (PERA), has been a fundamental part of industrial IT architecture since its development at Purdue University in the early 1990s (Figure 1). It acts as a bridge between OT and IT and provides a clear and structured method to ensure that automation systems communicate efficiently and securely with enterprise systems.

The model is divided into five levels, plus the Internet level, each of which represents different aspects and functions within an industrial network. Levels 0, 1, and 2 relate to direct production control. Level 0 includes sensors and actuators that interact directly with the physical environment to measure and control physical variables. Level 1 encompasses smart devices, such as programmable logic controllers (PLCs) that handle basic automation control tasks. Level 2 refers to higher level control systems that provide process controls and operational monitoring.

Level 3 deals with operations management and bridges the gap between direct production control and business applications. It is where production planning takes place and operational processes are managed. Level 4 takes care of enterprise management in the scope of enterprise resource planning (ERP), which is essential for handling inventory management, order management, and financial accounting. Level 5 represents the highest level of

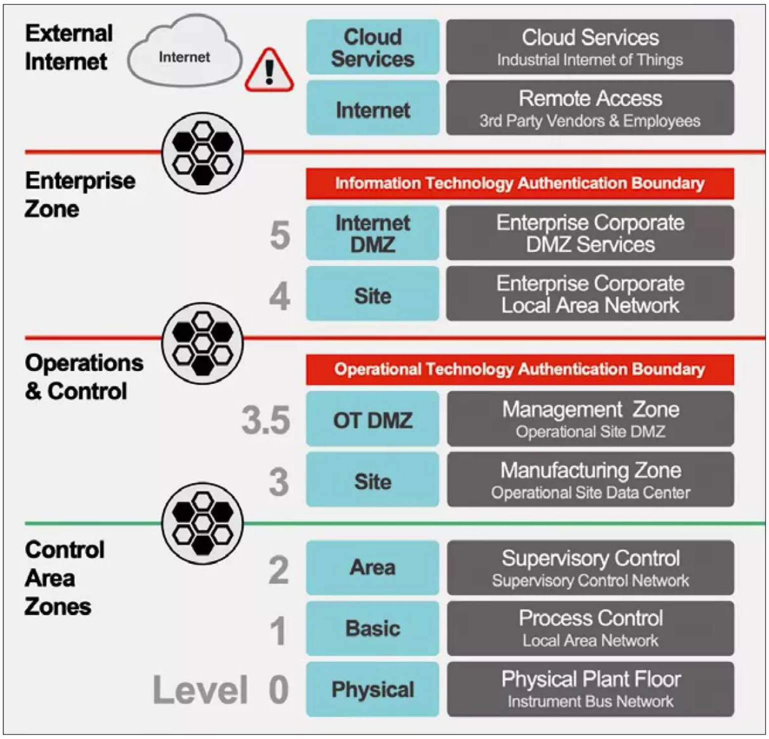

External Internet
Internet
Cloud Services
Cloud Services
Industrial Internet of Things
Internet
Remote Access
3rd Party Vendors & Employees
Enterprise Zone
Information Technology Authentication Boundary
5
Internet DMZ
Enterprise Corporate DMZ Services
4
Site
Enterprise Corporate Local Area Network
Operations & Control
Operational Technology Authentication Boundary
3.5
OT DMZ
Management Zone
Operational Site DMZ
3
Site
Manufacturing Zone
Operational Site Data Center
Control Area Zones
2
Area
Supervisory Control
Supervisory Control Network
1
Basic
Process Control
Local Area Network
Level 0
Physical
Physical Plant Floor
Instrument Bus Network

Figure 1: The five levels of the Purdue Model for industrial control systems. (Above: DMZ, demilitarized zone.)

corporate planning, where strategic decisions are made and analyses are carried out on the basis of aggregated data.

Above all these levels is the connection to the Internet, forming a critical interface to the outside world. On one hand, external networking enables the efficient use of cloud services and the exchange of data across company boundaries; on the other, it harbors increased risks in the form of potential cyberattacks.

Security Strategies

The clear structure and delineation of levels in the Purdue Model are crucial for the development of targeted security and control strategies. Giving each level its specific security protocols tailored to its needs ensures effective protection of the integrity of industrial control systems and company data alike and includes not just physical and network-based security, but also the implementation of data security campaigns and effective protection against unauthorized access to ensure a comprehensive security architecture.

This model puts organizations in a position to establish a robust, security-oriented structure that effectively supports both the operational and information technology aspects of their organization, which has never been so relevant as in our age of increasing digitalization and networking, where cyberthreats represent an ever-growing challenge.

Products and services that offer protection in accordance with the structured approach of the Purdue Model need to incorporate various technologies, as shown in Table 1.

FortiDeceptor

Fortinet is the organization behind FortiDeceptor [1], a specialized tool that uses honeypot technology to attract attackers by simulating vulnerable systems on the network and distract them from real targets. FortiDeceptor was developed specifically for use in OT environments and aims to detect and neutralize typical attack patterns that are common in industrial settings. By creating a controlled and secure environment, the product security teams observe and analyze the behavior of attackers without compromising physical systems or data. The insights into attack vectors and methods gained here enable organizations to adapt and strengthen their security strategies effectively.

One key feature is the ability not only to identify attacks, but also respond proactively. This defense supports integration into the Fortinet Security Fabric, which enables real-time data transfers and coordinated responses across a zoo of security systems that help shut down security gaps more quickly and mitigate potential damage.

The extension of the digital ceiling in smart facilities is a targeted example. By integrating fake IP cameras, alarm systems, and heating, ventilation,

Table 1: Purdue Model

Technology	Levels	Description
Perimeter protection	0, 1, 3, 4	The entire network is secured by strictly controlling and monitoring access from outside, which is particularly critical for the upper levels, where data is exchanged with external sources.
Layer 2 and 3 (L2/L3) NGFW segmentation	2, 3.5, 5, Internet	Next-generation firewalls (NGFWs) support advanced network segmentation, an essential factor for isolating the levels of the Purdue Model from each other. Segmentation is critical to prevent the spread of threats across the network.
Server protection	3, 4	Servers hosting critical, operational applications need to be protected. Malware, ransomware, and other types of threats must be tackled here.
Identity and access management and privileged access management		These systems regulate access to critical data and systems and ensure that only authorized persons have access, which is hugely important across all levels.
Threat protection (IDS/IPS)		Intrusion detection (IDS) and prevention systems (IPS) can be used anywhere on the network and are important for detecting and preventing real-time threats.
Application control	2, 3.5, 5	Here, access to applications is controlled and regulated on the basis of specified user identities and policies to prevent unwanted or threatening application activity.
Endpoint protection		Secure workstations and mobile devices prevent threats accessing the network via the endpoints.
Deception technology	2, 3, 4	Deception technologies mislead attackers and distract them from the actual targets, which strengthens the overall security architecture.
Advanced threat protection and sandboxing	2, 3, 4	Sandboxing technology isolates and analyzes suspicious files and processes in a secure environment, which is crucial to detecting and subsequently neutralizing advanced threats.
Network operations (NOC/SOC)		Network (NOCs) and security operations centers (SOCs) provide uninterrupted monitoring and response to security incidents, which is essential for early detection and a rapid response to security events.
Switching and WAP systems		Specialized switching systems (with PoE) and wireless access points (WAPs) are designed to connect OT environments securely to the network, which is critical to creating a robust infrastructure.
Ruggedized systems		Extreme conditions require robust systems that function reliably and have been specially developed for use in demanding OT environments such as DIN rail switch cabinets.

and air conditioning (HVAC) controls, FortiDeceptor can detect attackers at an early stage. The use of fake network infrastructures as decoys provides deep insights into the intruders' methods and behaviors, making it possible to act proactively and provide comprehensive protection.

The Challenge of Zero Trust

In the context of IT and OT, the zero trust security model, which is based on the principle of trusting nobody and nothing without appropriate verification, is becoming increasingly important. That said, implementing zero trust in OT environments poses problems, because these environments often contain outmoded technologies primarily designed for reliability and uptime rather than cybersecurity. Moreover, the OT landscape is often characterized by a lack of standardization, which adds complexity and harbors extensive potential for attacks.

For an effective implementation of zero trust in OT environments, you need to consider both the functionality of your industrial automation and control systems and the security-related aspects that play a role. It is also essential for the deployed zero trust technology to be compatible with legacy technology in OT environments, because certain components such as PLCs or HVAC systems might not be able to support the technologies or protocols required for full integration with zero trust.

Conclusions

The Purdue Model is a guide to abstracting a complex corporate network into a tangible image. The model's clear demarcation between individual components simplifies the planning and implementation of protection campaigns. The model also promotes communication and the distribution of responsibilities within the organization. The technical side has positive aspects, as well: Actions can be implemented at the boundaries between levels to monitor and control communication between the individual zones. ■

Info

[1] **FortiDeceptor:** [https://www.fortinet.com/products/fortideceptor]

Linux Magazine Subscription

Print and digital options
12 issues per year

▶ **SUBSCRIBE**
shop.linuxnewmedia.com

Expand your Linux skills:

- In-depth articles on trending topics, including Bitcoin, ransomware, cloud computing, and more!
- How-tos and tutorials on useful tools that will save you time and protect your data
- Troubleshooting and optimization tips
- News on crucial developments in the world of open source
- Cool projects for Raspberry Pi, Arduino, and other maker-board systems

Go farther and do more with Linux, subscribe today and never miss another issue!

Need more Linux?
Subscribe free to Linux Update

Our free Linux Update newsletter delivers insightful articles and tech tips to your inbox every week.

https://bit.ly/Linux-Update

Tools for hardware diagnostics under Windows

Put Through Their Paces

We look at one Windows on-board tool, five free tools, and a commercial tool to discover hardware installed on a Windows computer and determine whether it is still working properly. By Thomas Bär and Frank-Michael Schlede

Anyone looking for software that can be used to explore and check the hardware of a Windows computer could initially be overwhelmed by the sheer number of tools available. In this article, we look at a number of useful tools for this task, but we cannot claim this list is exhaustive because of the sheer number of tools on the market. Instead, we decided to test on-board resources and tools from the Windows environment, the open source community, and commercial programs that provide useful and practical support for controlling and managing hardware under Windows.

Windows Offerings

Both the older Windows versions from NT 4.0 and the current versions come with the `msinfo32` command-line program, which reports a first look of the machine hardware. The program offers a good overview of the available devices and hardware components and can be called up from the search box or directly at the command line. In the typical style of command-line programs, `msinfo32` can be called up with various options. For example, you can query the data of another system in the network with the command:

```
msinfo32 "/computer <computer name>"
```

Hardware information includes details about the processor and motherboard and a couple of other things in the program's system overview, but `msinfo32` delivers a mass of data related to software components, as well. The *Hardware Resources* entry in the sidebar of the System Information dialog contains detailed information on RAM and installed controllers. The *Components* category lists devices for CD, DVD, and hard drives. Whereas *Drives* means filesystems and their assignments, the *Disks* entry gives you information on the hard disks installed, such as their manufacturers and connections.

This information can also be found in the *System* section of the Windows settings on Windows 10 and 11 and on the latest Windows server versions. However, because searching here is a slow experience, `msinfo32` is definitely a good starting point if you do not want to install or use any third-party software on your system.

If you are familiar with PowerShell and have the skills to develop the right kind of scripts for your systems, you can of course also glean the desired information with scripts, while retaining the freedom to choose exactly what you need. To get started, try calling `Get-ComputerInfo`, which basically displays all the data from the call to `msinfo32` on the screen. Also you can use a number of cmdlets for specific hardware queries. If you are familiar with Windows Management Instrumentation (WMI) and Common Information Model (CIM), you can also run all the commands across the network on other systems. This ability means that you can create highly individual and very powerful scripts for hardware diagnostics.

CPU-Z

If you do not have the experience or time to use PowerShell options, you can find numerous free programs for system diagnostics that include both general-purpose tools that provide the most comprehensive details possible and tools that focus and report on specific critical hardware components. One of the best-known programs in this category is certainly CPU-Z [1], which gives you in-depth data on the computer's CPU along

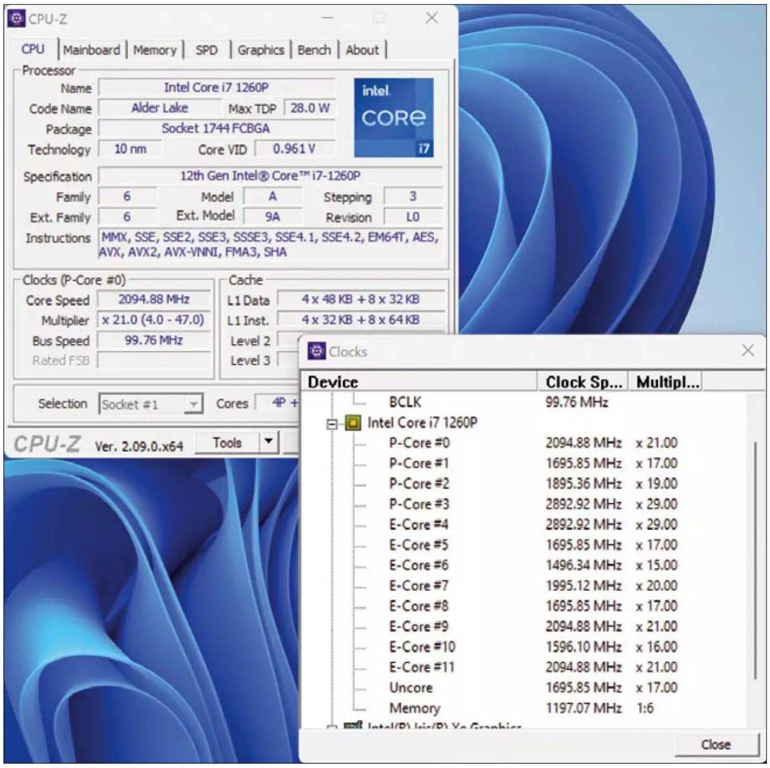
Lead Image © blas, Fotolia.com

with information on the mainboard and graphics card used (**Figure 1**). Although you will need to install the freeware program on the computer you want to examine first, the program is quite small, occupying less than 5MB of space on the hard drive. CPU-Z provides a wide range of information and gives you a very detailed overview of the CPU immediately after launching, including manufacturer details and information on the clock rate, the multiplier used, and the number of cores. Further tabs provide detailed information on the motherboard, graphics card, and main memory. The *SPD* (serial presence detect) tab is certainly special. It reports the memory type, size, timings, and module specifications the motherboard uses to configure access to the memory modules.

To test the speed of a computer, you can run a benchmark on the CPU (for a single thread or multithreads). This free software is definitely recommended, especially for admins in small businesses and office environments without a large IT team and without test equipment for what is normally a small number of computers. It is also helpful when it comes to taking an inventory of devices: A pull-down menu below the *Tools* tab at the bottom helps you create reports in HTML and text formats.

Sandra

Any review of well-known and free programs would definitely be incomplete without Sandra [2] by SiSoftware. This very versatile software package is available in various commercial versions. Besides the free Lite version, which we look at in more detail here, you can choose between a Professional Personal version for home use, through Professional Business and Engineer versions, to an Enterprise option. The Lite version is only intended for private use and currently supports systems from Windows 7 onward. If you want to use the Sandra software on a Windows server, you need the commercially licensed Engineer or Enterprise version.



Processor
Name: Intel Core i7 1260P
Code Name: Alder Lake
Max TDP: 28.0 W
Package: Socket 1744 FCBGA
Technology: 10 nm
Core VID: 0.961 V
Specification: 12th Gen Intel® Core™ i7-1260P
Family: 6
Model: A
Stepping: 3
Ext. Family: 6
Ext. Model: 9A
Revision: L0
Instructions: MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, EM64T, AES, AVX, AVX2, AVX-VNNI, FMA3, SHA

Clocks (P-Core #0)
Core Speed: 2094.88 MHz
Multiplier: x 21.0 (4.0 - 47.0)
Bus Speed: 99.76 MHz
Rated FSB:

Cache
L1 Data: 4 x 48 KB + 8 x 32 KB
L1 Inst.: 4 x 32 KB + 8 x 64 KB
Level 2:
Level 3:

Selection: Socket #1
Cores: 4P +

CPU-Z Ver. 2.09.0.x64

Device	Clock Sp...	Multipl...
BCLK	99.76 MHz	
Intel Core i7 1260P		
P-Core #0	2094.88 MHz	x 21.00
P-Core #1	1695.85 MHz	x 17.00
P-Core #2	1895.36 MHz	x 19.00
P-Core #3	2892.92 MHz	x 29.00
E-Core #4	2892.92 MHz	x 29.00
E-Core #5	1695.85 MHz	x 17.00
E-Core #6	1496.34 MHz	x 15.00
E-Core #7	1995.12 MHz	x 20.00
E-Core #8	1695.85 MHz	x 17.00
E-Core #9	2094.88 MHz	x 21.00
E-Core #10	1596.10 MHz	x 16.00
E-Core #11	2094.88 MHz	x 21.00
Uncore	1695.85 MHz	x 17.00
Memory	1197.07 MHz	1:6

Figure 1: CPU-Z focuses on data about the CPU, mainboard, and memory.

Even if the comparison list on the provider's website shows that many features are not available in the Lite version, this tool is still ideal for analyzing a Windows system's hardware. Remote services can also be enabled during the install. After completing the installation, Sandra Lite occupies just south of 200MB on your hard drive, which is surprisingly little in view of the broad feature set. One particular advantage of the tool is that the developers behind this project always look to support the latest and newest hardware. The R26 release (version 31.137), which became available in October 2023, can handle updates to Intel's generation 14 Core Raptor Lake processors and AMD's generation 4 Ryzen CPUs.

Sandra Lite combines a large number of different functions for analyzing and testing the computer under one interface. The functions are sorted by component and cover the CPU and RAM, the mainboard, hard drives, graphics cards, and many other devices. The options and views are presented in a clearly structured overview, featuring icons that show green if the feature is available on the device. Even the complete listing of a device in the Computer Management area provides very detailed information on the screen. To dig deeper, switch to the module in which you are interested; you can use the settings to specify in a targeted way the options you want to display.

Another focus of this program is on benchmarks that scrutinize a computer's performance. You can use benchmarks to check hardware components such as the CPU, graphics cards, or memory devices and compare the results with other systems once the test is complete. If you are looking to examine and test Windows hardware in detail, you will discover that Sandra is an ideal tool in this respect, and even the free version already has a mass of options up for grabs. A little patience is always advisable; the program performs the checks very thoroughly and can take a while to display the results on screen.

Speccy for Newcomers

Piriform (now part of Gen Digital), the maker of the well-known CCleaner software, also offers Speccy [3], a software tool for system analysis. Besides the free version, an extended professional license offers premium support. After a brief analysis, the overview displayed after installation and startup shows on screen a mass of data relating to the system and its hardware. Clicking on the respective components takes you to further, more detailed information. The software also displays values in real time (e.g., the bus speed and CPU core temperatures) and is therefore useful for continuous monitoring.

You can export all the acquired system data in the form of a report in TXT or XML format. As a special feature, the program also lets you upload a report anonymously to the Piriform website. You can then pass on the associated URL (e.g., to exchange information with other users). Unfortunately, this operation only works for the full set of data. Just as with the reports, you cannot choose to export only certain data; instead, you need to edit the text or XML file.

Speccy is particularly suitable for less experienced admins because of its simple and clear-cut operation, although it does give you a useful collection of data relating to the hardware of a Windows system. Unfortunately, the software seems to be under sporadic development – or none at all. According to the copyright, the latest version is from 2020 and the software struggles to recognize some newer CPUs, such as an Intel Core i7 1260P from the Alter Lake family. It simply states that the CPU is an "Intel Processor," although you can still read the correct string for the chip specification in the details.

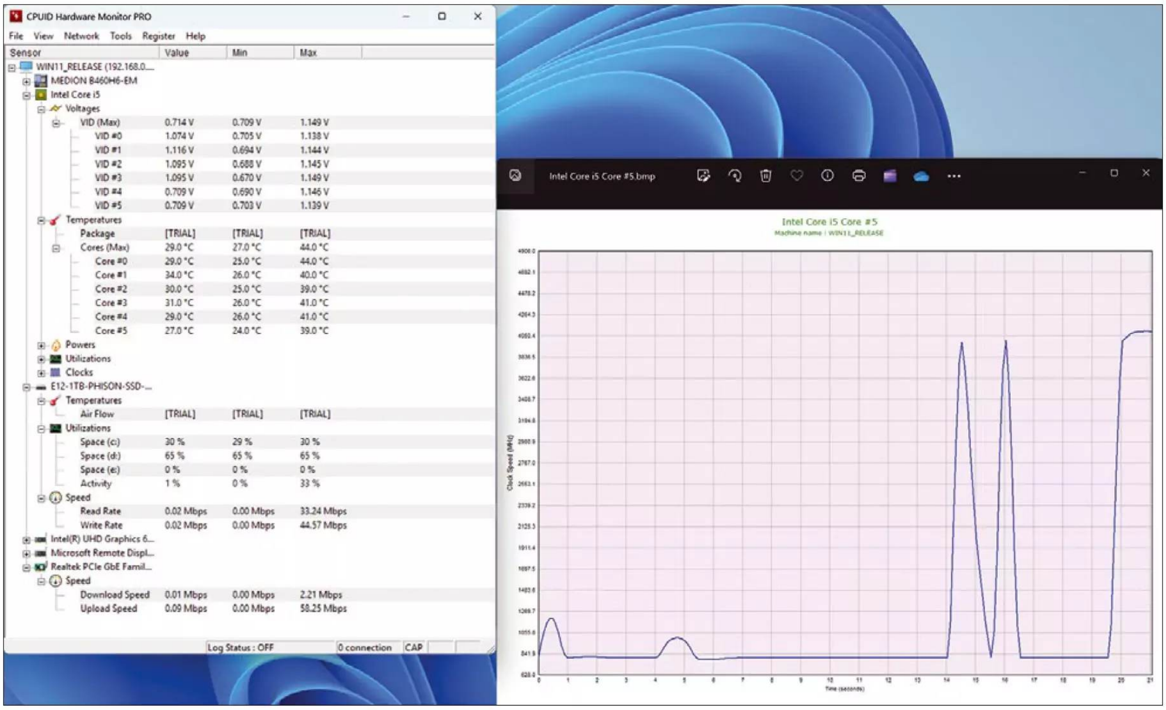
HWMonitor

HWMonitor [4] comes from the same developers as CPU-Z and specializes in reading and monitoring the parameters of a motherboard for a simple but very detailed live view of the temperature and voltage values of the CPU, fans, and hard drives (Figure 2).

The free version can only be used on a local system. In contrast, the commercial Pro version can access the sensor data of other Windows and Android systems over TCP/IP. This version is available for a 30-day trial with minor functional restrictions and then requires licensing to give you a year of updates and unlimited use for approximately $22 (EUR22, £18). This extended version also lets you save the log data and displays the results as simple graphics. The interface is more flexible in the Pro version and you can then also store sensors in the system trays on your Windows computers. That said, the standard version of HWMonitor is fine for reading the live data from the various system sensors. The software is accurate and reliable, but only lets you save the data in a text file.

CrystalDiskInfo

The free CrystalDiskInfo [5] tool is, without a doubt, one of the best known programs for system analysis. Whereas many of the programs presented here record and display the values of the hard drives installed in a system as a nice add-on, CrystalDiskInfo is designed exclusively for disk diagnostics,



```
Sensor                          Value        Min          Max
WIN11_RELEASE (192.168.0....
  MEDION B460H6-EM
  Intel Core i5
    Voltages
      VID (Max)                 0.714 V      0.709 V      1.149 V
        VID #0                  1.074 V      0.705 V      1.138 V
        VID #1                  1.116 V      0.694 V      1.144 V
        VID #2                  1.095 V      0.688 V      1.145 V
        VID #3                  1.095 V      0.670 V      1.149 V
        VID #4                  0.709 V      0.690 V      1.146 V
        VID #5                  0.709 V      0.703 V      1.139 V
    Temperatures
      Package                   [TRIAL]      [TRIAL]      [TRIAL]
      Cores (Max)               29.0 °C      27.0 °C      44.0 °C
        Core #0                 29.0 °C      25.0 °C      44.0 °C
        Core #1                 34.0 °C      26.0 °C      40.0 °C
        Core #2                 30.0 °C      25.0 °C      39.0 °C
        Core #3                 31.0 °C      26.0 °C      41.0 °C
        Core #4                 29.0 °C      26.0 °C      41.0 °C
        Core #5                 27.0 °C      24.0 °C      39.0 °C
    Powers
    Utilizations
    Clocks
  E12-1TB-PHISON-SSD-...
    Temperatures
      Air Flow                  [TRIAL]      [TRIAL]      [TRIAL]
    Utilizations
      Space (c:)                30 %         29 %         30 %
      Space (d:)                65 %         65 %         65 %
      Space (e:)                0 %          0 %          0 %
      Activity                  1 %          0 %          33 %
    Speed
      Read Rate                 0.02 Mbps    0.00 Mbps    33.24 Mbps
      Write Rate                0.02 Mbps    0.00 Mbps    44.57 Mbps
  Intel(R) UHD Graphics 6...
  Microsoft Remote Displ...
  Realtek PCIe GbE Famil...
    Speed
      Download Speed            0.01 Mbps    0.00 Mbps    2.21 Mbps
      Upload Speed              0.09 Mbps    0.00 Mbps    58.25 Mbps

Log Status : OFF      0 connection      CAP
```

Figure 2: HWMonitor impresses with a very focused display that concentrates on the live data from the various hardware sensors.

delivering a full set of data and values related to hard and solid state drives (**Figure 3**). Besides the current S.M.A.R.T. status, the program displays the temperature, speed, write and read error rates, storage space, buffer sizes, firmware version, serial number, and operating hours of the hard drive.

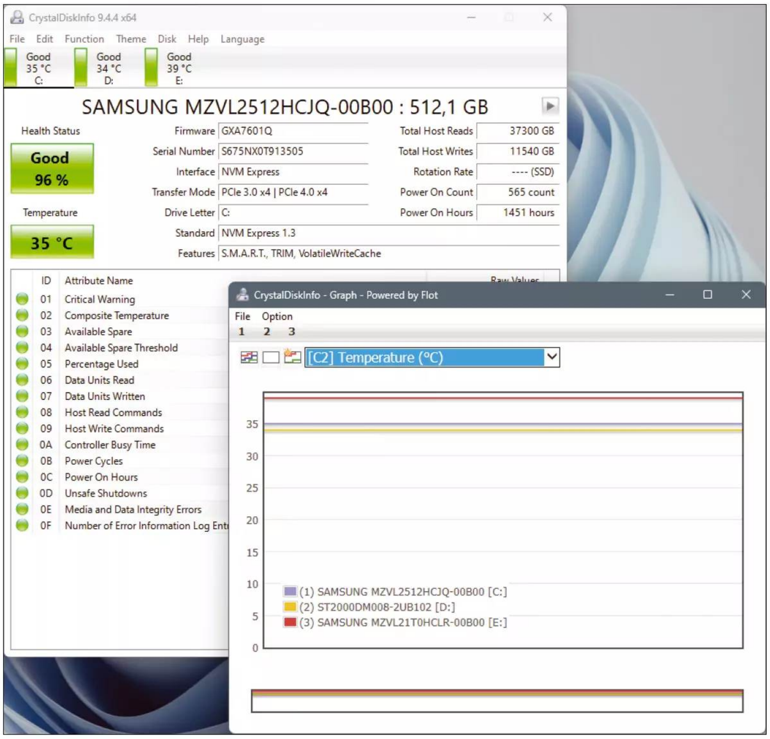
In addition to supplying accurate data, CrystalDiskInfo uses a color scheme to provide information about the status of a hard drive. By default, the most important information about the hard disks is shown in green or blue if no problems are detected. This combination of simple, easy-to-understand information with detailed S.M.A.R.T. data makes CrystalDiskInfo an ideal tool for admins who want to analyze and monitor hard drives.

AIDA64

AIDA64 [6] is only available in various commercial versions. AIDA64 Extreme (around $60, EUR60, £38) is a diagnostic and benchmark tool for home users. The AIDA64 Engineer license ($200, EUR200) offers, among other things, to automate processes by entering data at the command line. The Network Audit and Business versions specialize in network and IT statistics, respectively, but AIDA64 Extreme is the core product; you can download it directly from the website as a 30-day trial version (you need to email the manufacturer for all other versions).

The provider specifically points out that the software can monitor more than 250 sensors in real time. An *OSD* (on-screen display) entry in the menu draws a plain-looking window on your screen that continuously shows the values of free memory, mass storage, and temperatures and voltages of the motherboard and CPU. These values can also be displayed by clicking on an icon in the system tray or using various other devices, such as the LCD display on some keyboards.

AIDA64 is certainly a good recommendation for IT professionals who keep a constant eye on the data of individual systems.


CrystalDiskInfo 9.4.4 x64
File Edit Function Theme Disk Help Language
Good 35 °C C:
Good 34 °C D:
Good 39 °C E:
CrystalDiskInfo - Graph - Powered by Flot
File Option
1 2 3
[C2] Temperature (°C)
35
30
25
20
15
10
5
0
(1) SAMSUNG MZVL2512HCJQ-00B00 [C:]
(2) ST2000DM008-2UB102 [D:]
(3) SAMSUNG MZVL21T0HCLR-00B00 [E:]

SAMSUNG MZVL2512HCJQ-00B00 : 512,1 GB

Health Status: Good 96 %
Temperature: 35 °C
Firmware: GXA7601Q
Serial Number: S675NX0T913505
Interface: NVM Express
Transfer Mode: PCIe 3.0 x4 | PCIe 4.0 x4
Drive Letter: C:
Standard: NVM Express 1.3
Features: S.M.A.R.T., TRIM, VolatileWriteCache
Total Host Reads: 37300 GB
Total Host Writes: 11540 GB
Rotation Rate: ---- (SSD)
Power On Count: 565 count
Power On Hours: 1451 hours

ID	Attribute Name
01	Critical Warning
02	Composite Temperature
03	Available Spare
04	Available Spare Threshold
05	Percentage Used
06	Data Units Read
07	Data Units Written
08	Host Read Commands
09	Host Write Commands
0A	Controller Busy Time
0B	Power Cycles
0C	Power On Hours
0D	Unsafe Shutdowns
0E	Media and Data Integrity Errors
0F	Number of Error Information Log Entr

Figure 3: Virtually no other tool offers the same depth of information about mass storage devices as CrystalDiskInfo.

Conclusions

In small businesses, Windows onboard tools such as `msinfo32` are a good starting point. Because the software also provides values from other computers in the network, you can quickly gain a general overview of your Windows computers. If a system then requires a more detailed analysis, you can use free programs such as CPU-Z during troubleshooting to discover precisely which CPU type and which memory modules are installed. If you want to go deeper, you can benchmark your systems with programs such as Sandra Lite and benefit from optimization tips. The more sophisticated tools, of which AIDA64 is an example, impress with support for the latest hardware and display values and data that might not be covered by freeware. Even extreme stress tests can be carried out with this category of analysis programs. Ultimately, no panacea can solve all problems, and we can only advise that you test the various applications on the devices on your own networks first. In our opinion, it is essential for the software to support reporting – although most free programs we looked at can do this without breaking a sweat. ■

Info

[1] CPU-Z: [https://www.cpuid.com/softwares/cpu-z.html]
[2] Sandra Lite: [https://sisoft-sandra-lite.en.softonic.com/]
[3] Speccy: [https://www.ccleaner.com/speccy]
[4] HWMonitor: [https://www.cpuid.com/softwares/hwmonitor.html]
[5] CrystalDiskInfo: [https://crystalmark.info/en/software/crystaldiskinfo/]
[6] AIDA64: [https://www.aida64.com/products]

Author

Thomas Bär has worked in IT since the late 1990s and as a freelance IT journalist since the early 2000s.

Frank-Michael Schlede was an editor and editor-in-chief for various publishing houses for more 25 years. Since 2010 he has been a freelance journalist, often in collaboration with Thomas Bär.

Getting started with AI

Quick Launch

A great way to start writing code with AI is to use Keras, an open source easy-to-learn library that can use multiple frameworks.

By Jeff Layton

I wanted to start learning about artificial intelligence (AI) and writing code, but I wasn't sure where to start. I didn't want to write in C++ or use a directed acyclic graph (DAG) or some other computer-science-centric language. I wanted to be able to write something straightforward, without complication, that allowed me to define clearly what I wanted to do.

The early years of AI saw the emergence of a wide range of frameworks. Most of them used Python, but all of them had their own language for reading the data, defining the model, and solving weights, which exposed my fears of learning a new "language." Which framework should I select and why? Was one a clear leader?

I didn't see any pointers that said, "go this way," and I didn't see any articles that explained the trade-offs, but I did read about Keras [1], and that is where I started. (Note that I can't write about the things I do or learn in my day job, but this subject I learned before joining.)

A Brief History of Keras

Keras was developed by François Chollet in 2014 out of necessity for an open source implementation of recurrent neural networks (RNNs) and the long short-term memory (LSTM) models that train them. Keras is Python based, and was first released in March 2015. It developed a following fairly quickly because the models were popular at the time.

Up to version 2.3, Keras supported multiple frameworks (back ends) that included TensorFlow, Microsoft Cognitive Toolkit (commonly referred to as CNTK), Theano, and PlaidML. By version 2.4, Keras only supported TensorFlow. In version 3.0 and subsequent versions, Keras once again supported multiple frameworks, including TensorFlow, PyTorch, and JAX.

Fundamentally, Keras abstracts away the details of the back-end frameworks so that you call functions from Keras to build and train models. The application programming interface (API) is consistent regardless of the back end and, in many cases, is much easier to use than those in a specific framework.

This last point is important. You don't have to learn TensorFlow or PyTorch or JAX to use the framework effectively. You can learn Keras and use all three frameworks. Then, if you want, you can pick one framework to learn more in depth, having learned the concepts and developed ideas with Keras.

Keras and VGG16

Getting started with Keras is not difficult. Rather than use the MNIST [2] dataset of 60,000 grayscale images as an example, I'll use a VGG16 [3] model as the example. It is a larger model than MNIST and more useful, so it's more realistic. Moreover, it is easy to understand because it is a sequential convolutional neural network (CNN).

The VGG16 model was developed as part of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014. Karen Simonyan and Andrew Zisserman from the Visual Geometry Group (Department of Engineering Science, University of Oxford) showcased their model at the contest and introduced it in a paper [4]. You can see that the name of the group, Visual Geometry Group, gave rise to the name (VGG).

The model is focused on object detection and classification tasks. The model comprises:

- 13 convolutional layers
- 3 fully connected layers
- 5 pooling layers

The pooling layers are not counted in the total number of layers, resulting in 16 total layers (hence the name). You can see the layout of the layers in a thorough blog post on VGG16 [5].

I won't go into the details of the model, but I want to show you an example of Keras-based code that implements the model and trains it on a very simple dataset of images of cats and dogs.

System Assumptions

I'll assume your system is Linux or macOS just because I haven't tested Windows.

I'll also assume you have at least 16GB of memory and a GPU that is compatible with Keras with at least

Lead Image © Lightwise, 123RF.com

4GB of dedicated video memory. I used an NVIDIA GPU.

You should have a recent version of Python installed on your system – I used Python 3.8.10 – and a specific Python environment built for testing, which I'll discuss. I don't recommend using your base environment, although I've been known to do that. (It's simple enough to erase Python 3 and reinstall it or just force a reinstall.)

Keras Install and Back-End Framework

Before you start writing code, you have to install Keras and a matching back-end framework on your system. The Keras website says you can install it with PyPI:

```
$ pip install --upgrade keras
```

This command should install the GPU version of Keras as well. If you like, you can check the Keras version with the commands:

```
$ python3
>>> import keras
>>> print(keras.__version__)
```

You also need to install one of the back ends – TensorFlow, PyTorch, or JAX – on your system. From the Keras URL you can go to the page for whichever back end you want to use. I used TensorFlow. The PyPI command to install TensorFlow is simple:

```
$ pip install tensorflow
```

This command should also install the GPU version of TensorFlow. Be sure to check the version installed:

```
$ python3
>>> import tensorflow as tf
>>> print(tf.__version__)
```

It doesn't make too much difference, but if you are curious, I used TensorFlow 2.9.2 and Keras 2.9.0. The TensorFlow version is a bit old; 2.16.1 is the latest as of this writing, but I already had it installed. My Keras is also a bit old. I think Keras 3.6 is the latest, and my version isn't even 3.x. I haven't tested the code with Keras 3.x yet, so your mileage may vary if you go that route.

CIFAR-10

The model and dataset I use is CIFAR-10 [6]. It is a very common dataset in computer vision and classification, but the images are low resolution (32x32). However, the low resolution makes the training go faster.

CIFAR-10 stands for Canadian Institute for Advance Research with 10 outputs or classes. The 10 image classes are:

- Airplane
- Car
- Bird
- Cat
- Deer
- Dog
- Frog
- Horse
- Ship
- Truck

Python Modules

To start writing code, open your favorite editor in the directory where you downloaded and unzipped the data. The first few lines in the code should load the needed modules (**Listing 1**). This code is probably not strictly Pythonic, but I don't write Pythonic code. I like to solve problems, not worry about proper syntax. If it works, it works.

The entire Keras library is imported (`import keras`) followed by the specific keras functions to be used.

Starting the Code

Start the code by reading the data with a built-in Keras function:

```
(train_images, train_labels), ➥
  (test_images, test_labels) = ➥
  cifar10.load_data()
```

This line uses a predefined Keras dataset routine named `cifar10`. It loads the dataset and, in this case, splits the data into two parts (`train_images`, `train_labels`), a tuple of the images and their corresponding labels that will be used to train the model. The labels specify the correct classification of the image. The second set of data (`test_images`, `test_labels`) is used to validate the model to check whether it works on non-training data. If you only check the model with the training data, you run the risk of overtraining your model, which will perform poorly on non-training data.

The next part of the code (**Listing 2**) standardizes the data and converts the labels to categories. The first two lines divide the number of possible pixels to standardize the data. The next three lines convert the labels to category data that can be used in training and checking the state of the model (validation).

Building the Model

This next step of constructing the VGG16 model is really fun because it shows off the power of Keras. The model is sequential starting with the first layer, then moving to the second layer, and so on, down to the final layer, which is the output. The first two lines of the model are:

```
model = Sequential()
model.add(layers.Conv2D(32, (3,3), ➥
  padding='same', activation='relu', ➥
  input_shape=(32,32,3)))
```

The first line initializes the model with the `Sequential` class from Keras (`keras.models.Sequential`). The next line is the first layer of the model, which is also referred to as the input layer, which is a convolutional layer [7] (`Conv2D`) with an `input_shape` of 32x32 – two-dimensional (2D) image – and three channels (RGB values). The first layer

Listing 1: Importing Modules

```
import os

os.environ["KERAS_BACKEND"] = "tensorflow"

import keras
from keras.datasets import cifar10
from keras.models import Sequential
from keras import datasets, layers, models
from keras import regularizers
from keras.layers import Dense, Dropout, BatchNormalization

import numpy as np
```

uses 32 convolutional filters (first argument) with a kernel size of 3x3 (second argument).

To decode this a bit more, a convolutional filter is applied to the image. The filter is 3x3 and scans the entire 32x32 image in some fashion such as left to right and top to bottom. Thirty-two of these filters run across the image. The values of the 32 3x3 filters are parameters of the model that are to be adjusted during training. The input to the filter is an image from the collection of training images, and the output is a convolved image that goes through an `activation` function and is passed to the next layer. In this case, it is the rectified linear unit (ReLU) function (`relu`).

As a reminder, the parameters of this layer are the values in each 3x3 convolution filter for all 32 filters. These are the weights that are to be trained (initially they are just random values).

The `padding` argument specifies that padding is even to the left/right or up/down of the image, so the output size is the same size as the input (i.e., the image has not changed size), which is relevant to the extra image pixels that are around the edges of the input image.

Finally, the `activation` function for this layer, the ReLU function, is very common in image models, but I won't discuss it in this article.

Next is a normalization [8] layer:

```
model.add(layers.BatchNormalization())
```

This function applies a transformation so that the mean of the output is around zero and the standard deviation is around one.

One thing you should note is that Keras takes care of making sure all of the output dimensions from the input layer with the 2D convolution match the inputs to the batch normalization layer.

Next is another convolutional layer that is identical to the first layer, except you don't need to specify the input size because Keras ensures that the input size of this layer matches the output size from the previous layer:

```
model.add(layers.Conv2D(32, (3,3), ➘
  padding='same', activation='relu'))
```

After this second 2D convolution layer, you have another batch normalization layer followed by a new layer type, `MaxPooling2D`:

```
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D➘
  (pool_size=(2,2)))
```

A max pooling layer has a pool size, 2x2 in this case, that is used to scan the entire input image (left to right, top to bottom). The pool is like a filter. Within this pool, the largest value is used for the output and downsizes the image from the previous layer. In this case, it reduces it by two.

A `strides` argument in the max pooling layer defines how the pool window moves across the image. In this case, it will move two pixels to the right/left or down/up, which allows you to overlap pools (e.g., to retain some knowledge from other pools) and allows the pooling layer to downsize the image more or less than the pool size.

Again, note that Keras takes care of matching the output from the previous layer to the input for the next layer. If there is a fundamental problem, Keras gives an error and stops.

The max pooling layer is followed by a new layer called a `Dropout` layer:

```
model.add(layers.Dropout(0.3))
```

This layer randomly sets certain inputs to `0` with a specified frequency – in this case `0.3` (30%) – which helps prevent overfitting of the training data; however, it does not affect the overall generalization of the model or the training of the model to the test dataset.

The first six layers of the model (call it a block) are shown in **Listing 3**.

The next six layers of the model (**Listing 4**) are the same except for some small changes:

- `input_shape` does not need to be specified in the first 2D convolutional layer.
- Convolutional filters number 64 instead of 32.
- The dropout rate has been increased to `0.5` (50%).

A third block (**Listing 5**) is the same, except for some small parameter changes as before (number of convolutional filers, etc).

Only one more small block of model layers remains. Recall that the model is processing 2D images, which doesn't work well when categorizing images: How do you map them? The Keras `Flatten` function converts a 2D image to a 1D vector:

```
model.add(layers.Flatten())
```

This layer reshapes the data from the previous layer, which is 2D, to a one-dimensional vector that can be used in "classic" fully connected layers. In this way, you convert an image into something you can ultimately use as output to classify the image.

At this point, you add another layer, which is referred to as a dense layer. Think of the classic model of a neural network, which has layers of neurons in a column (i.e., a 1D vector). This dense layer is going to process the inputs with a ReLU activation function:

```
model.add(layers.Dense➘
  (128, activation='relu'))
```

Listing 2: Standardizing Data

```
# Standardizing (255 is the total number of pixels an image can have)
train_images = train_images / 255
test_images = test_images / 255

# One hot encoding the target class (labels)
num_classes = 10
train_labels = keras.utils.to_categorical(train_labels, num_classes)
test_labels = keras.utils.to_categorical(test_labels, num_classes)
```

Listing 3: First Block

```
model.add(layers.Conv2D(32, (3,3), padding='same', activation='relu', input_shape=(32,32,3)))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(32, (3,3), padding='same', activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.Dropout(0.3))
```

This particular model then adds a `BatchNormalization` layer followed by a `Dropout` layer:

```
model.add(layers.BatchNormalization())
model.add(layers.Dropout(0.5))
```

Finally, the last layer is a dense layer where the output number of neurons is `10`, which matches the number of classes of images:

```
model.add(layers.Dense(num_classes, ➘
  activation='softmax')) # num_classes = 10
```

Because this is the last layer, you need to use a `softmax` activation function. You can read more about the `softmax` function **[9]** and why `softmax` is used for classification models **[10]**.

An important consideration is that when an image is processed through the model, you get numbers for all 10 classes that are the probabilities that the processed image falls into any of those classes. You will never get an image with a 100 percent (`1.0`) probability in a specific class and a zero in all other classes. Neural networks generalize; they don't give you a 100 percent specific answer. However, if you look at all of the probabilities, you can probably tell to which class the image belongs because it has the larger value.

If you have an image in which two or more classes have almost the same probability, the model is having a difficult time determining to which class the image belongs. If this happens, you likely need to gather more testing data and create a larger model.

Listing 4: Second Block

```
model.add(layers.Conv2D(64, (3,3), padding='same', activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(64, (3,3), padding='same', activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.Dropout(0.5))
```

Listing 5: Third Block

```
model.add(layers.Conv2D(128, (3,3), padding='same', activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(128, (3,3), padding='same', activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.Dropout(0.5))
```

Remember that neural networks can be fooled by images just like humans. The entire model should look like **Listing 6**. Next, you need to compile the model, which builds the code that TensorFlow (or whatever back end you chose) runs for the model.

Compiling the Model

The next step is to compile the model **[11]**:

```
model.compile(optimizer='adam', ➘
  loss=keras.losses.categorical_ ➘
 crossentropy, metrics=['accuracy'])
```

Keras configures the overall learning process, specifically the `optimizer`, the *loss* function, and the `metrics`. The optimizer to be used is `adam` **[12]**, a stochastic gradient descent optimizer that makes estimation of the first and second order moments (gradients). For the model to be trained, the difference between the test image class (100% probability of a specific class and 0% probability of all other classes) and the results of the model is collected to produce a `loss`. For a specific image, the `loss` is the error between the test image probabilities and the model output probabilities, which is used by the optimizer to determine the new parameters. Typically, these parameters are the weights in the model for the next iterations. For this problem the `loss` function is defined as a categorical cross-entropy function **[13]**.

Train!

At this point, you can train the model, which means that all the training images are passed through the model and the `loss` function is computed. The `adam` algorithm then computes the changes that need to be made to the model parameters (weights) and applies them. This process is called the backpropagation step.

After the parameters have been updated with the new values, the process repeats until the model has converged, which can be when the *loss* function you defined when the model was compiled stops changing very much, the changes in the parameters are very small, or both. This point is referred to as the "stopping criteria." This process sounds like a loop, doesn't it? However, you don't have to write the details of the training loops, including computing the `loss` function and the updates, to the model parameters. Keras has

Listing 6: The Model

```
model = Sequential()

model.add(layers.Conv2D(32, (3,3), padding='same', activation='relu', input_
  shape=(32,32,3)))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(32, (3,3), padding='same', activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.Dropout(0.3))

model.add(layers.Conv2D(64, (3,3), padding='same', activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(64, (3,3), padding='same', activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.Dropout(0.5))

model.add(layers.Conv2D(128, (3,3), padding='same', activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(128, (3,3), padding='same', activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.Dropout(0.5))

model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(num_classes, activation='softmax'))    # num_classes = 10
```

a method or function that does all of this for you named `fit`:

```
history = model.fit(train_images, ↩
  train_labels, batch_size=64, ↩
  epochs=20, validation_data=↩
  (test_images, test_labels))
```

An iteration is referred to as an `epoch`, and this training loop only runs `20` epochs. Just change the value for `epochs` to run as many iterations as you want. Notice that you pass in the validation data to the function. It will run the test images through the model to check how well it's learning. The `batch_size=64` tells the `fit` function to take the images in groups of 64 and run them through the model. It will compute the gradients for the `adam` optimizer in the group of 64. The subject of batch size has been the topic of several papers. A small batch size usually means convergence will be slow, but it also frees up more memory because not as many images are stored at once. A large batch size means more memory usage and can result in early convergence.

Sample Output

I won't bore you with all the output, but I do want to share some of it.

Listing 7: First Three Epochs

```
================
== TensorFlow ==
================

NVIDIA Release 24.08-tf2 (build 106933591)
TensorFlow Version 2.16.1
Container image Copyright (c) 2024, NVIDIA CORPORATION & AFFILIATES. All rights reserved.
Copyright 2017-2024 The TensorFlow Authors.  All rights reserved.

Various files include modifications (c) NVIDIA CORPORATION & AFFILIATES.  All rights reserved.

This container image and its contents are governed by the NVIDIA Deep Learning Container License.
By pulling and using the container, you accept the terms and conditions of this license:
https://developer.nvidia.com/ngc/nvidia-deep-learning-container-license

NOTE: CUDA Forward Compatibility mode ENABLED.
  Using CUDA 12.6 driver version 560.35.03 with kernel driver version 535.161.08.
  See https://docs.nvidia.com/deploy/cuda-compatibility/ for details.

Epoch 1/100
782/782 ???????????????????? 21s 12ms/step - accuracy: 0.3156 - loss: 2.1379 - val_accuracy: 0.4105 -
  val_loss: 1.7636
Epoch 2/100
782/782 ???????????????????? 2s 3ms/step - accuracy: 0.5481 - loss: 1.2612 - val_accuracy: 0.5852 -
  val_loss: 1.2059
Epoch 3/100
139/782 ???????????????????? 1s 3ms/step - accuracy: 0.6390 - loss: 1.0173
```

Listing 7 shows the output for the first three epochs. Note that `val_accurancy` and `val_loss` are the accuracy and loss for the validation data, not the training data.

Summary

Keras is a great place to start your journey in AI. When you write your code in Keras, you can choose whichever back-end framework you want. Keras has much more capability, such as checking for convergence and stopping, writing checkpoints during training, summarizing the model, writing callback functions called by the `fit` function at various times during training, and more. You can also plot the training history from the results in the `history` variables, (i.e., the output from the `fit` function). Keras is great for experimenting and learning about various hyperparameters, which are the values you specify when creating the model, compiling the model, or running the training. Simple examples include the number of convolutional filters used with specific layers, the dropout rate, the activation function, and so on. There is no real science to selecting these hyperparameters, and you might have to try several options to get some feel for improving the accuracy of the model or improving convergence. Keras has a very large set of examples [14] for a wide range of topics, such as transformer-based examples to learn about how these popular building blocks work in generative AI examples, time series examples, audio examples, graph examples, structured data examples, and style transfer examples, as well as a nice section on generative examples [15] that go beyond the transformer examples. ■

Info

[1] Keras: [https://kera.io/]

[2] MNIST: [https://keras.io/examples/vision/mnist_convnet/]

[3] Beginner's Guide to VGG16 Implementation in Keras: [https://builtin.com/machine-learning/vgg16]

[4] Very Deep Convolutional Networks for Large-Scale Image Recognition: [https://arxiv.org/pdf/1409.1556]

[5] Everything you need to know about VGG16: [https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918]

[6] CIFAR-10: [https://en.wikipedia.org/wiki/CIFAR-10]

[7] Convolutional layer: [https://en.wikipedia.org/wiki/Convolutional_layer]

[8] Normalization layer: [https://keras.io/api/layers/normalization_layers/batch_normalization/]

[9] softmax Function: [https://en.wikipedia.org/wiki/Softmax_function]

[10] softmax Classification: [https://www.pinecone.io/learn/softmax-activation/]

[11] Compile the model: [https://saturncloud.io/blog/why-you-need-to-compile-your-keras-model-before-using-modelevaluate/]

[12] adam: [https://keras.io/api/optimizers/adam/]

[13] Categorical cross-entropy loss function: [https://vitalflux.com/keras-categorical-cross-entropy-loss-function/]

[14] Keras code examples: [https://keras.io/examples/]

[15] Generative examples: [https://keras.io/examples/generative/]

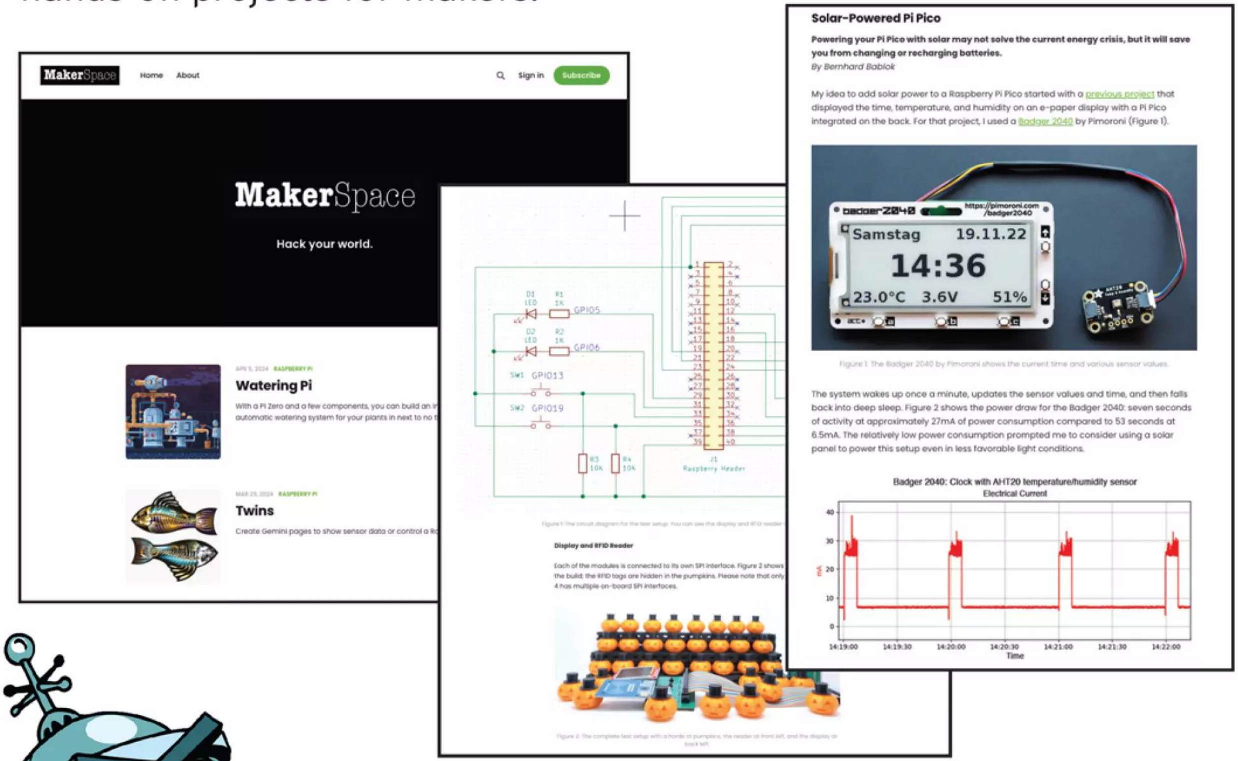
The Author

Jeff Layton has been in the HPC business for over 30 years (starting when he was 4 years old). When he's not grappling with a stubborn systemd script, he's looking for deals for his home cluster. His twitter handle is @JeffdotLayton.

MakerSpace-Online

New Maker Content Every Week

At MakerSpace, we are all about technology you can use to build your own stuff. Our goal is to help you turn your ideas into reality with hands-on projects for makers.



MakerSpace Home About Sign in Subscribe

MakerSpace

Hack your world.

RASPBERRY PI

Watering Pi

With a Pi Zero and a few components, you can build an i
automatic watering system for your plants in next to no t

RASPBERRY PI

Twins

Create Gemini pages to show sensor data or control a R

Display and RFID Reader

Each of the modules is connected to its own SPI interface. Figure 2 shows
the build; the RFID tags are hidden in the pumpkins. Please note that only
4 has multiple on-board SPI interfaces.

Solar-Powered Pi Pico

Powering your Pi Pico with solar may not solve the current energy crisis, but it will save you from changing or recharging batteries.

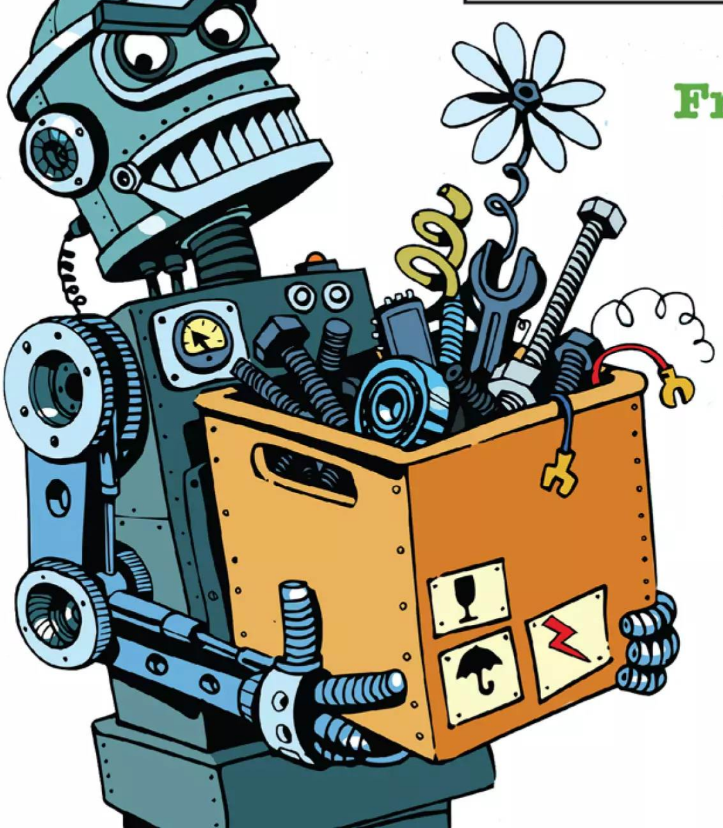
By Bernhard Bablok

My idea to add solar power to a Raspberry Pi Pico started with a previous project that displayed the time, temperature, and humidity on an e-paper display with a Pi Pico integrated on the back. For that project, I used a Badger 2040 by Pimoroni (Figure 1).

Figure 1: The Badger 2040 by Pimoroni shows the current time and various sensor values.

The system wakes up once a minute, updates the sensor values and time, and then falls back into deep sleep. Figure 2 shows the power draw for the Badger 2040: seven seconds of activity at approximately 27mA of power consumption compared to 53 seconds at 6.5mA. The relatively low power consumption prompted me to consider using a solar panel to power this setup even in less favorable light conditions.

D1 LED
R1 1K
GPIO5
D2 LED
R2 1K
GPIO6
SW1 GPIO13
SW2 GPIO19
R3 10K
R4 10K
J1
Raspberry Header
badger2040
https://pimoroni.com /badger2040
Samstag 19.11.22
14:36
23.0°C 3.6V 51%
Badger 2040: Clock with AHT20 temperature/humidity sensor
Electrical Current
mA
40
30
20
10
0
14:19:00
14:19:30
14:20:00
14:20:30
14:21:00
14:21:30
14:22:00
Time



Fresh Content, Delivered

Subscribe now and join the MakerSpace community. You'll stay up-to-date when new content is published.

Join Now!

https://makerspace-online.com

The best performance yet

Superhero

Unveiling little-known superpowers of microSD cards.

By Federico Lucifredi

Recently, I analyzed the performance of new storage options on Raspberry Pi 5 boards [1], focusing on the Pi 5's USB bus improvements (the USB 3 ports are now independently capable of 5Gbps each) and testing the even faster new PCIe expansion hat with NVMe drives. Only hinted at is that the SD card interface itself has received some sophisticated upgrades for the fifth Pi series. Not only do USB and NVMe storage options outclass any benchmark that could be run on SD cards, they are also more reliable, with SD cards a notoriously finicky storage solution. Despite all this, the convenience of SD cards is not to be outdone, and the vast majority of users choose to use the format to power their single-board computers (SBCs), making any advance in the area truly noteworthy.

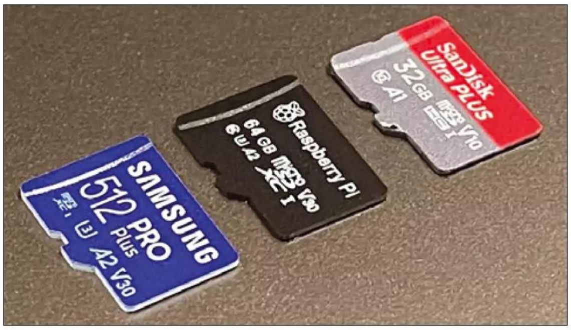


Figure 1: TF or SD, choose your trademark, but it is all the same to me.

SD Card Diagnostics

Because of the number of issues caused by low-performing SD cards in the past, the Raspberry Pi team released a validation tool [10] to diagnose whether a given card meets the SBC's requirements. Found under the Pi menu as *Accessories | Raspberry Pi Diagnostics* (Figure 2), after installing with

```
sudo apt install agnostics
```

(the name is an engineer's pun on diagnostics and has nothing to do with religion), the tool validates a card's compliance with the SD card performance requirements for the A1 class.

These tests should be run on a freshly formatted card, because the state of garbage collection (TRIM) could affect the tests. Additionally, the benchmarks are run multiple times, with their results averaged to minimize the effect of spurious results. Figure 2 shows a 32GB SanDisk Ultra Plus posting 91.5MBps read and 20.2MBps write sequential throughput. The 4K Random I/O averaged 4,010 read and 1,019 write I/O operations per second (IOPS).

A handy tool maintained by the community for quickly benchmarking storage is provided by PiBenchmarks [11] through a direct Bash download that will give pause to folks with the proper respect for security:

```
sudo curl https://raw.githubusercontent.com/TheRemote/PiBenchmarks/master/Storage.sh | sudo bash
```

The script runs a battery of standard benchmarks and conveniently summarizes the results (Figure 3); with your permission, it uploads the full configuration and results to an online reference site. You can see my own result in Figure 4.

The Same, but Different

TransFlash (TF) cards were developed and marketed by Motorola and SanDisk back in 2004 [2], then adopted by the SD Association as the microSD [3] format specification in 2005. The microSD cards were functionally the same as those original TF cards. Use of the TF trademark does not require vendors to pay fees, which is why it has suddenly become popular in some quarters to use the TF monicker again (Figure 1). Whether you refer to it as SD or TF, the Raspberry Pi 5 is now capable of supporting Command Queuing (CQ) when used in combination with a card supporting the extended standard (SDXC). CQ mode was introduced in 2017 with version 6.0 of the SD specification [4], but although the Linux kernel had been supporting CQ mode for eMMC storage for some time, the SD media driver was not capable of the same feat until recently. It fell to Jonathan Bell to bring Linux up to speed to recover this insofar missed performance opportunity [5], delivering updated drivers that were included in Raspberry Pi OS and are now part of the development kernel since March 2024 [6].

The PCI SD Host Controller bridge (SDHCI) section of the SD controller manages communication between the host and the SD card. The Command Queuing Host Controller Interface (CQHCI) extends the SDHCI with new hardware that takes over when a compatible card is loaded. In this mode, read and write operations can proceed asynchronously. The card can reorder operations for performance

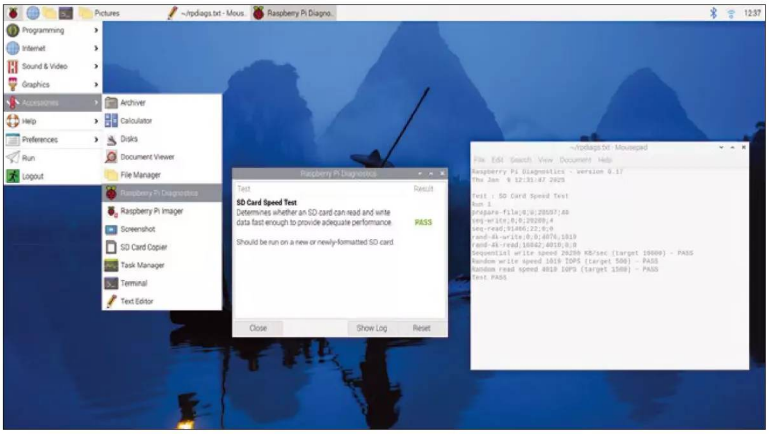
Lead Image © Lucy Baldwin, 123RF.com

optimization – which, in combination with optional support for write buffering, delivers a performance multiplier for random I/O performance. The new driver can put compatible cards in CQ mode, after which a different set of SD commands becomes available to the host. Although the new cards will still perform really well in older Raspberry Pi boards, support for CQ mode strictly requires a Pi 5.

Playing Your Cards

With the write performance of vintage SD cards a major issue, SD card choices made a big difference historically. As of 2023, though, all Class 10 devices (or better) I could purchase made the mark in my tests for this very column **[7]**, relegating that problem to history's dustbin – to the point that Jeff Geerling no longer needs to maintain his revered SD card benchmarks tracking page **[8]**. However, not many cards for sale today support CQ mode operation – SanDisk Extreme models are reported to do so, as are Raspberry Pi's own cards, which are manufactured by Longsys **[9]**. Surprises while testing CQ functionality with A2-rated cards sold by vendors best left unnamed led Raspberry Pi down the road of marketing its own branded cards to avoid a sequel to the "will this card work well with my Pi" saga of lore (see the "SD Card Diagnostics" box).

Raspberry Pi-branded microSDHCs (or microSDXC for the larger sizes; SDHC is limited to 32GB) are available in 32GB and 64GB capacities, with an announced 128GB size. They achieved the most stringent A2, U3, and V30 speed class certifications, as well as the lesser C10. They meet the SD-6.1 specification and support Command Queueing, DDR50, and SDR104 bus speeds **[12]**. The introduction of an SDR104 SD drive has essentially doubled the performance of SD cards meeting the UHS-I spec when comparing Raspberry Pis 4 and 5. The original, default UHS-I data rate of 100MHz delivered a throughput of 50MBps – commonly referred to as DDR50. The additional 208MHz mode is also required for microSDHC and microSDXC


Pictures
~/rpdiags.txt - Mous...
Raspberry Pi Diagno...
12:37
Programming
Internet
Sound & Video
Graphics
Accessories
Help
Preferences
Run
Logout
Archiver
Calculator
Disks
Document Viewer
File Manager
Raspberry Pi Diagnostics
Raspberry Pi Imager
Screenshot
SD Card Copier
Task Manager
Terminal
Text Editor
Raspberry Pi Diagnostics
Test
Result
Close
Show Log
Reset
~/rpdiags.txt - Mousepad
File
Edit
Search
View
Document
Help

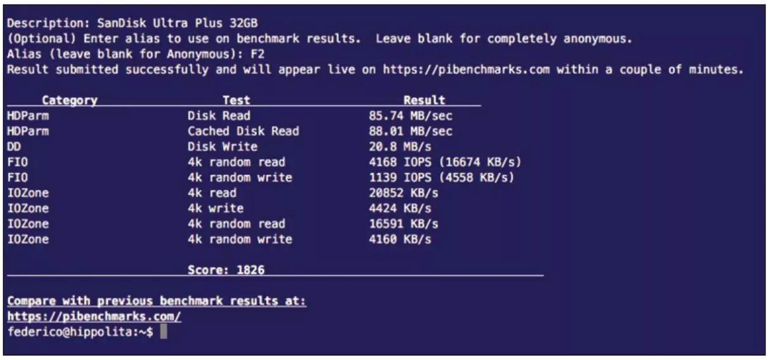
SD Card Speed Test
Determines whether an SD card can read and write data fast enough to provide adequate performance — PASS

Should be run on a new or newly-formatted SD card

```
Raspberry Pi Diagnostics - version 0.17
Thu Jan  9 12:31:47 2025

Test : SD Card Speed Test
Run 1
prepare-file;0;0;20597;40
seq-write;0;0;20289;4
seq-read;91466;22;0;0
rand-4k-write;0;0;4076;1019
rand-4k-read;16042;4010;0;0
Sequential write speed 20289 KB/sec (target 10000) - PASS
Random write speed 1019 IOPS (target 500) - PASS
Random read speed 4010 IOPS (target 1500) - PASS
Test PASS
```

Figure 2: My SanDisk Ultra Plus boot volume handily meets the A1 performance class targets laid out by the SD card diagnostics tool.



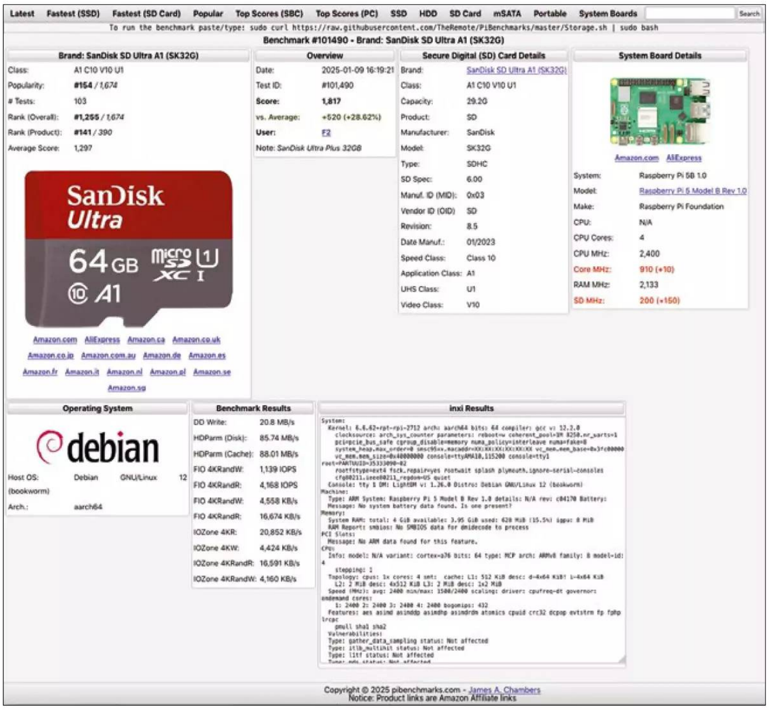
```
Description: SanDisk Ultra Plus 32GB
(Optional) Enter alias to use on benchmark results.  Leave blank for completely anonymous.
Alias (leave blank for Anonymous): F2
Result submitted successfully and will appear live on https://pibenchmarks.com within a couple of minutes.

     Category                  Test                      Result
HDParm                    Disk Read                 85.74 MB/sec
HDParm                    Cached Disk Read          88.01 MB/sec
DD                        Disk Write                20.8 MB/s
FIO                       4k random read            4168 IOPS (16674 KB/s)
FIO                       4k random write           1139 IOPS (4558 KB/s)
IOZone                    4k read                   20852 KB/s
IOZone                    4k write                  4424 KB/s
IOZone                    4k random read            16591 KB/s
IOZone                    4k random write           4160 KB/s

                          Score: 1826

Compare with previous benchmark results at:
https://pibenchmarks.com/
federico@hippolita:~$ 
```

Figure 3: The PiBenchmarks.com `storage.sh` is a favorite of the Raspberry Pi community.


Latest
Fastest (SSD)
Fastest (SD Card)
Popular
Top Scores (SBC)
Top Scores (PC)
SSD
HDD
SD Card
mSATA
Portable
System Boards
Search
SanDisk Ultra
64GB microSD XC I
A1
Amazon.com
AliExpress
Amazon.ca
Amazon.co.uk
Amazon.co.jp
Amazon.com.au
Amazon.de
Amazon.es
Amazon.fr
Amazon.it
Amazon.nl
Amazon.pl
Amazon.se
Amazon.sg
debian
Amazon.com
AliExpress

To run the benchmark paste/type: sudo curl https://raw.githubusercontent.com/TheRemote/PiBenchmarks/master/Storage.sh | sudo bash

Benchmark #101490 - Brand: SanDisk SD Ultra A1 (SK32G)

Brand: SanDisk SD Ultra A1 (SK32G)
Class: A1 C10 V10 U1
Popularity: #154 / 1,674
Tests: 103
Rank (Overall): #1,255 / 1,674
Rank (Product): #141 / 390
Average Score: 1,297

Overview
Date: 2025-01-09 16:19:21
Test ID: #101,490
Score: 1,817
vs. Average: +520 (+28.62%)
User: F2
Note: *SanDisk Ultra Plus 32GB*

Secure Digital (SD) Card Details
Brand: SanDisk SD Ultra A1 (SK32G)
Class: A1 C10 V10 U1
Capacity: 29.2G
Product: SD
Manufacturer: SanDisk
Model: SK32G
Type: SDHC
SD Spec: 6.00
Manuf. ID (MID): 0x03
Vendor ID (OID): SD
Revision: 8.5
Date Manuf.: 01/2023
Speed Class: Class 10
Application Class: A1
UHS Class: U1
Video Class: V10

System Board Details
System: Raspberry Pi 5B 1.0
Model: Raspberry Pi 5 Model B Rev 1.0
Make: Raspberry Pi Foundation
CPU: N/A
CPU Cores: 4
CPU MHz: 2,400
Core MHz: 910 (+10)
RAM MHz: 2,133
SD MHz: 200 (+150)

Operating System
Host OS: Debian GNU/Linux 12 (bookworm)
Arch.: aarch64

Benchmark Results
DD Write: 20.8 MB/s
HDParm (Disk): 85.74 MB/s
HDParm (Cache): 88.01 MB/s
FIO 4KRandW: 1,139 IOPS
FIO 4KRandR: 4,168 IOPS
FIO 4KRandW: 4,558 KB/s
FIO 4KRandR: 16,674 KB/s
IOZone 4KR: 20,852 KB/s
IOZone 4KW: 4,424 KB/s
IOZone 4KRandR: 16,591 KB/s
IOZone 4KRandW: 4,160 KB/s

inxi Results

Copyright © 2025 pibenchmarks.com - James A. Chambers
Notice: Product links are Amazon Affiliate links

Figure 4: The online report matching the test run shown in Figure 3.

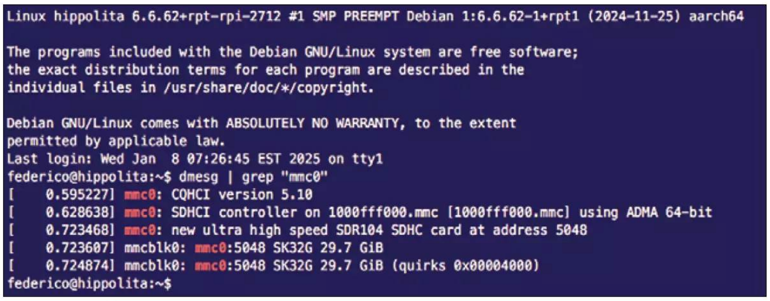
cards meeting the UHS-I spec and can deliver up to 104MBps of I/O.

Benchmarks

Raspberry Pi reports the cards demonstrate significant improvement in random I/O, sustaining 5,000 read and 2,000 write 4K IOPS in SDR104 mode on a Pi 5 [13], with the older Pi 4 achieving a still very good 3,200 reads and 1,200 writes in DDR50 [1] mode.

However, as the unofficial motto of this column goes, there are only lies, damn lies, and benchmarks; therefore, you have to see this for yourself. Pulling my lab's testing Pi 5, I imaged my pre-existing configuration on a new 64GB Raspberry Pi-branded card and ran a full system upgrade to insure I have the latest drivers. To bring your OS version up to the latest and greatest bits, use:

```
sudo apt update && sudo apt full-upgrade
```

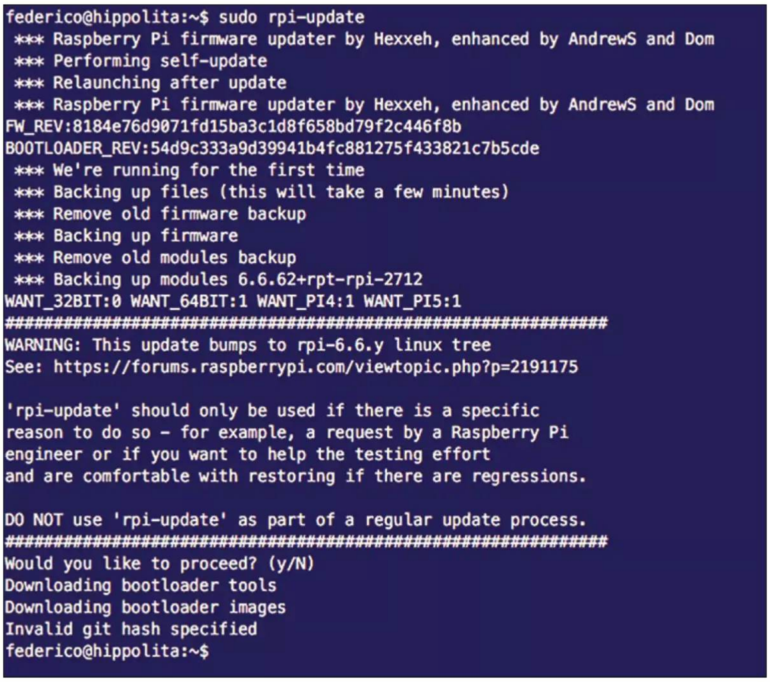


```
Linux hippolita 6.6.62+rpt-rpi-2712 #1 SMP PREEMPT Debian 1:6.6.62-1+rpt1 (2024-11-25) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Jan  8 07:26:45 EST 2025 on tty1
federico@hippolita:~$ dmesg | grep "mmc0"
[    0.595227] mmc0: CQHCI version 5.10
[    0.628638] mmc0: SDHCI controller on 1000fff000.mmc [1000fff000.mmc] using ADMA 64-bit
[    0.723468] mmc0: new ultra high speed SDR104 SDHC card at address 5048
[    0.723607] mmcblk0: mmc0:5048 SK32G 29.7 GiB
[    0.724874] mmcblk0: mmc0:5048 SK32G 29.7 GiB (quirks 0x00004000)
federico@hippolita:~$
```

Figure 5: CQHCI detected, but with the SDHCI controller used for the tests in the previous *ADMIN* article [1].

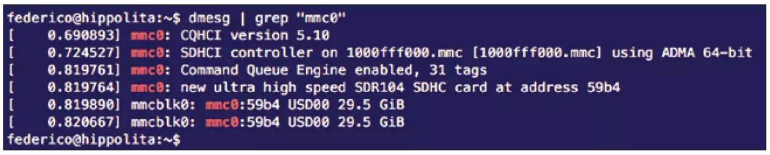


```
federico@hippolita:~$ sudo rpi-update
 *** Raspberry Pi firmware updater by Hexxeh, enhanced by AndrewS and Dom
 *** Performing self-update
 *** Relaunching after update
 *** Raspberry Pi firmware updater by Hexxeh, enhanced by AndrewS and Dom
FW_REV:8184e76d9071fd15ba3c1d8f658bd79f2c446f8b
BOOTLOADER_REV:54d9c333a9d39941b4fc881275f433821c7b5cde
 *** We're running for the first time
 *** Backing up files (this will take a few minutes)
 *** Remove old firmware backup
 *** Backing up firmware
 *** Remove old modules backup
 *** Backing up modules 6.6.62+rpt-rpi-2712
WANT_32BIT:0 WANT_64BIT:1 WANT_PI4:1 WANT_PI5:1
#############################################################
WARNING: This update bumps to rpi-6.6.y linux tree
See: https://forums.raspberrypi.com/viewtopic.php?p=2191175

'rpi-update' should only be used if there is a specific
reason to do so - for example, a request by a Raspberry Pi
engineer or if you want to help the testing effort
and are comfortable with restoring if there are regressions.

DO NOT use 'rpi-update' as part of a regular update process.
#############################################################
Would you like to proceed? (y/N)
Downloading bootloader tools
Downloading bootloader images
Invalid git hash specified
federico@hippolita:~$
```

Figure 6: Installation process for the Raspberry Pi development kernel.



```
federico@hippolita:~$ dmesg | grep "mmc0"
[    0.690893] mmc0: CQHCI version 5.10
[    0.724527] mmc0: SDHCI controller on 1000fff000.mmc [1000fff000.mmc] using ADMA 64-bit
[    0.819761] mmc0: Command Queue Engine enabled, 31 tags
[    0.819764] mmc0: new ultra high speed SDR104 SDHC card at address 59b4
[    0.819890] mmcblk0: mmc0:59b4 USD00 29.5 GiB
[    0.820667] mmcblk0: mmc0:59b4 USD00 29.5 GiB
federico@hippolita:~$
```

Figure 7: CQ engine detected – and activated by its driver.

In my previous article, throughput rose to 85.42MBps [1], compared with 43.53MBps [7] in my tests with 32GB SanDisk Ultra Plus cards (V10 A1 Class 10 HC I), which highlighted the benefit of an SDR104 bus when compared with the older DDR50. However, CQ had not been enabled for those tests, because the SanDisk Ultra Plus cards do not appear to carry a CQ engine (see Figure 5).

There is more fuel to add to this fire! I installed the latest development kernel,

```
sudo rpi-update
```

on the Longsys card (Figure 6), see the "Upgrading a Raspberry Pi" box. After a reboot,

```
dmesg | grep "mmc0"
```

I confirmed the presence of a CQ engine and its activation (Figure 7). Up to 32 requests can be processed simultaneously, which can really benefit random I/O performance. Testing the resulting setup confirms this prediction (Figure 8), with the card far exceeding its A2 class rating, posting 6,055 read and 2,563 write IOPS (the class requires 4,000 and 2,000, respectively). Final confirmation of the CQ engine's contribution is confirmed by de-activating it – adding

```
dtparam=sd_cqe=off
```

to `/boot/firmware/config.txt` and rebooting will do the trick, which results in substantially unchanged sequential performance, but a drop-off of 26 percent in read and 34 percent in write I/O, respectively (Figure 9). Paying more for a fancy SD card finally pays off! ■

Info

[1] "Pushing Raspberry Pi Storage to Its Limit" by Federico Lucifredi, *ADMIN*, issue 84, 2024: [https://www.admin-magazine.com/Archive/2024/84/Pushing-Raspberry-Pi-storage-to-its-limit]

[2] SanDisk releases new memory cards: [https://www.cnet.com/tech/tech-industry/sandisk-releases-new-memory-cards/]

[3] Wikipedia, SD Card – microSD: [https://en.wikipedia.org/wiki/SD_card#microSD]

[4] SD Standard Overview, The SD Association: [https://www.sdcard.org/developers/sd-standard-overview/]

[5] "Raspberry Pi SD Cards" by Jonathan Bell, *The MagPi*, issue 147, November 2024: [https://magpi.raspberrypi.com/issues/147]

[6] Raspberry Pi Linux kernel: [https://github.com/raspberrypi/linux/blob/rpi-6.8.y/drivers/mmc/core/sd.c#L1521]

[7] "Finding the Fastest SD Cards for the Raspberry Pi" by Federico Lucifredi, *ADMIN*, issue 76, 2023: [https://www.admin-magazine.com/Archive/2023/76/Finding-the-fastest-SD-cards-for-the-Raspberry-Pi]

[8] microSD card benchmarks: [http://www.pidramble.com/wiki/benchmarks/microsd-cards]

[9] "Raspberry Pi Launches Own-Brand, High-Performance microSD Cards" by Gareth Halfcree, *Hackster.io*, October 2023: [https://www.hackster.io/news/raspberry-pi-launches-own-brand-high-performance-microsd-cards-6a8dd8bc3253]

[10] Raspberry Pi SD card speed test: [https://www.raspberrypi.com/news/sd-card-speed-test/]

[11] storage.sh by PiBenchmarks: [https://raw.githubusercontent.com/TheRemote/PiBenchmarks/master/Storage.sh]

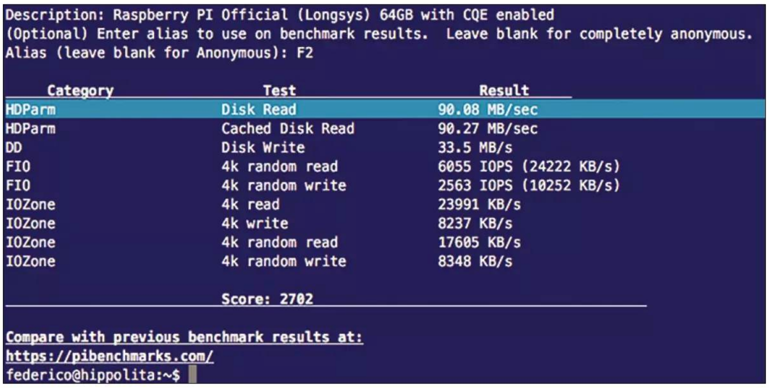
[12] Raspberry Pi SD cards: [https://www.raspberrypi.com/products/sd-cards/]

[13] Wikipedia, SD card – UHS-I: [https://en.wikipedia.org/wiki/SD_card#UHS-I]

[14] Manage software packages with APT: [https://www.raspberrypi.com/documentation/computers/os.html#manage-software-packages-with-apt]

[15] Upgrade your firmware: [https://www.raspberrypi.com/documentation/computers/os.html#rpi-update]

[16] rpi-eeprom-update: [https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#rpi-eeprom-update]



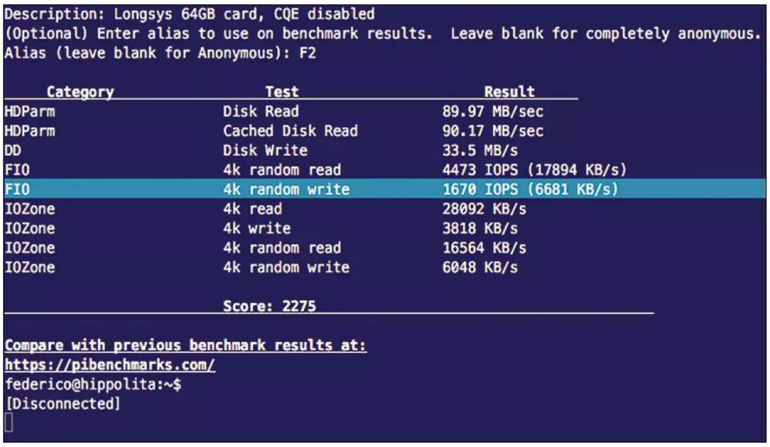
```
Description: Raspberry PI Official (Longsys) 64GB with CQE enabled
(Optional) Enter alias to use on benchmark results.  Leave blank for completely anonymous.
Alias (leave blank for Anonymous): F2

     Category                Test                    Result
HDParm                  Disk Read                 90.08 MB/sec
HDParm                  Cached Disk Read          90.27 MB/sec
DD                      Disk Write                33.5 MB/s
FIO                     4k random read            6055 IOPS (24222 KB/s)
FIO                     4k random write           2563 IOPS (10252 KB/s)
IOZone                  4k read                   23991 KB/s
IOZone                  4k write                  8237 KB/s
IOZone                  4k random read            17605 KB/s
IOZone                  4k random write           8348 KB/s

                        Score: 2702

Compare with previous benchmark results at:
https://pibenchmarks.com/
federico@hippolita:~$
```

Figure 8: A remarkable showing: 90MBps of sequential throughput and more than 6,000 random reads.



```
Description: Longsys 64GB card, CQE disabled
(Optional) Enter alias to use on benchmark results.  Leave blank for completely anonymous.
Alias (leave blank for Anonymous): F2

     Category                Test                    Result
HDParm                  Disk Read                 89.97 MB/sec
HDParm                  Cached Disk Read          90.17 MB/sec
DD                      Disk Write                33.5 MB/s
FIO                     4k random read            4473 IOPS (17894 KB/s)
FIO                     4k random write           1670 IOPS (6681 KB/s)
IOZone                  4k read                   28092 KB/s
IOZone                  4k write                  3818 KB/s
IOZone                  4k random read            16564 KB/s
IOZone                  4k random write           6048 KB/s

                        Score: 2275

Compare with previous benchmark results at:
https://pibenchmarks.com/
federico@hippolita:~$
[Disconnected]
```

Figure 9: Disabling the CQ engine had a severe effect on asynchronous I/O, but almost no effect on sequential throughput.

Upgrading a Raspberry Pi

Let us count the ways (Table 1). The routine avenue to upgrading a Raspberry Pi is through Debian's Advanced Package Tool (`apt`) [14] and is performed with a twinned pair of commands, as shown in the first entry of Table 1. Although package upgrades might not require a `reboot`, firmware and bootloader upgrades most definitely do. My personal strategy is to use a separate SD card for `rpi-update` kernels to revert changes by re-imaging.

Table 1: Upgrading a Raspberry Pi

Command	Description
`sudo apt update && sudo apt full-upgrade`	Update the repository metadata first and then perform the upgrade to the latest packages available.
`sudo rpi-update`	Installs the latest pre-release version of the kernel, its modules, device tree, and VideoCore firmware into an existing Raspberry Pi OS install. Pre-release software is in no way guaranteed to work and could, in a worst case, render your board permanently inoperable. Because of the risk of bricking a device, this option is limited to those feeling adventurous and willing to take chances with truly cutting-edge software [15].
`sudo rpi-eeprom-update`	Installs the latest bootloader and its configuration. This command is usually wrapped into higher level upgrade processes, but it can also be run interactively if a bootloader upgrade is sought specifically [16].

ADMIN
Network & Security

NEWSSTAND

Order online:
https://bit.ly/ADMIN-library

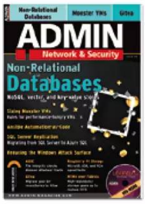
ADMIN is your source for technical solutions to real-world problems. Every issue is packed with practical articles on the topics you need, such as: security, cloud computing, DevOps, HPC, storage, and more! Explore our full catalog of back issues for specific topics or to complete your collection.

#84 – November/December 2024

Non-Relational Databases

NoSQL databases manage data unsuitable for table-based relational databases and deliver efficiencies in scaling and high availability.

On the DVD: Ubuntu 24.04 LTS Server "Noble Numbat"

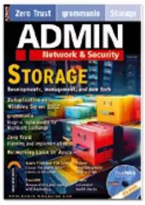

ADMIN
Network & Security
Non-Relational
Databases

#83 – September/October 2024

Storage

Improved management, suitable drivers, standardized protocol structures, and advancements in future-proof hardware and software lead to more durable, more manageable, and easier-to-repair storage.

On the DVD: TrueNAS SCALE 24.04 "Dragonfish"


ADMIN
Network & Security
STORAGE

#82 – July/August 2024

Sovereign Cloud Stack

SCS liberates your data centers from monopolistic operations and companies beholden to out-country laws and regulations.

On the DVD: Kali Linux 2024.2

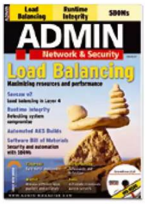
ADMIN
Network & Security
Sovereign
Cloud Stack

#81 – May/June 2024

Load Balancing

Load balancing on heavily frequented networks improves performance, availability, security, scalability, and the ability to handle peak loads.

On the DVD: SystemRescue 11.01

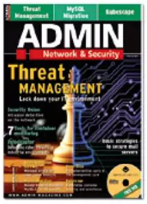

ADMIN
Network & Security
Load Balancing

#80 – March/April 2024

Threat Management

Digital infrastructures are vulnerable to all kinds of attacks. You need strategies and tools to detect and defend.

On the DVD: openSUSE Leap 15.5

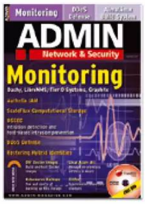

ADMIN
Network & Security
Threat
MANAGEMENT

#79 – January/February 2024

Monitoring

This issue takes a deep dive into monitoring solutions for your IT infrastructure, including Dashy, LibreNMS, Tier 0 systems, and Graphite.

On the DVD: FreeBSD 14.0


ADMIN
Network & Security
Monitoring

WRITE FOR US

Admin: Network and Security is looking for good, practical articles on system administration topics. We love to hear from IT professionals who have discovered innovative tools or techniques for solving real-world problems.

Tell us about your favorite:

- Interoperability solutions
- Practical tools for cloud environments
- Security problems and how you solved them
- Ingenious custom scripts
- Unheralded open source utilities
- Windows networking techniques that aren't explained (or aren't explained well) in the standard documentation

We need concrete, fully developed solutions: installation steps, configuration files, examples – we are looking for a complete discussion, not just a "hot tip" that leaves the details to the reader.

If you have an idea for an article, send a 1-2 paragraph proposal describing your topic to: *edit@admin-magazine.com*.

Authors

Amber Ankerholz	6
Thomas Bär	82
Erik Bärwaldt	12
Dr. Wilhelm Greiner	72
Ken Hess	3
Jeff Layton	9, 86
Martin Gerhard Loschwitz	24, 38, 44
Federico Lucifredi	92
Gerd Pflüger	78
Lee Phillips	62
Dr. Holger Reibold	50
Frank-Michael Schlede	82
Tim Schürmann	18
Koen Vervloesem	56
Kevin Wittmer	30

Contact Info

Editor in Chief
Joe Casad, jcasad@linuxnewmedia.com

Managing Editors
Rita L Sooby, rsooby@linuxnewmedia.com
Lori White, lwhite@linuxnewmedia.com

Senior Editor
Ken Hess

Localization & Translation
Ian Travis

News Editor
Amber Ankerholz

Copy Editors
Amy Pettle, Aubrey Vaughn

Layout
Dena Friesen, Lori White

Cover Design
Dena Friesen, Illustration based on graphics by thvideo, 123RF.com

Advertising
Jessica Pryor, jpryor@linuxnewmedia.com

Publisher
Brian Osborn

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
4840 Bob Billings Parkway, Ste 104
Lawrence, KS 66049 USA

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxnewmedia.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linuxnewmedia.com
www.admin-magazine.com

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the DVD provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2025 Linux New Media USA, LLC.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media unless otherwise stated in writing.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Nuremberg, Germany by be1druckt GmbH.

Distributed by Seymour Distribution Ltd, United Kingdom

ADMIN (Print ISSN: 2045-0702, Online ISSN: 2831-9583, USPS No: 347-931) is published bimonthly by Linux New Media USA, LLC, and distributed in the USA by Asendia USA, 701 Ashland Ave, Folcroft PA. January/February 2025. Application to Mail at Periodicals Postage Prices is pending at Philadelphia, PA and additional mailing offices. POSTMASTER: send address changes to Linux Magazine, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Represented in Europe and other territories by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Available Starting
April 4

ADMIN 86

Our next issue will be packed with all the great content you expect from *ADMIN*. Here are a few of the upcoming articles:

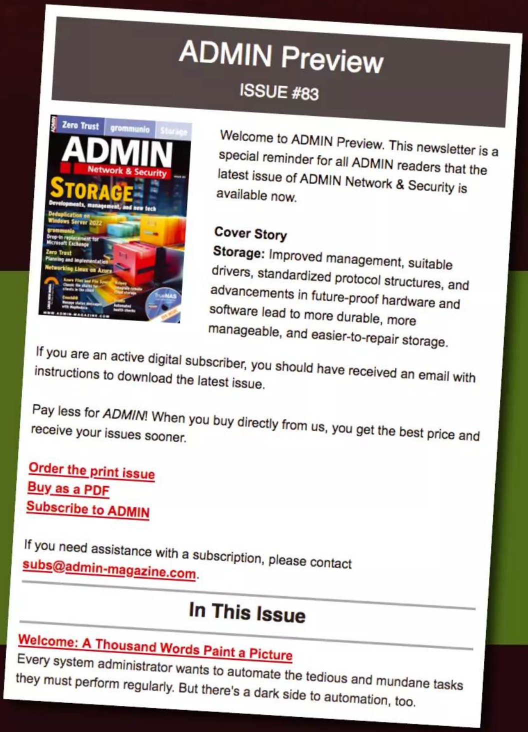
- **Volcano Gang Scheduler**
- Maltrail
- Wireguard Portal
- Managing dotfiles
- And much more!

Please note: Articles could change before the next issue.

BE THE FIRST TO SEE WHAT'S NEXT

Subscribe free to the *ADMIN* Preview newsletter and get a sneak peek at every article included in the next issue of *ADMIN*.

Sign up today at https://bit.ly/admin-preview



ADMIN Preview

ISSUE #83

Welcome to ADMIN Preview. This newsletter is a special reminder for all ADMIN readers that the latest issue of ADMIN Network & Security is available now.

Cover Story

Storage: Improved management, suitable drivers, standardized protocol structures, and advancements in future-proof hardware and software lead to more durable, more manageable, and easier-to-repair storage.

If you are an active digital subscriber, you should have received an email with instructions to download the latest issue.

Pay less for *ADMIN*! When you buy directly from us, you get the best price and receive your issues sooner.

Order the print issue
Buy as a PDF
Subscribe to ADMIN

If you need assistance with a subscription, please contact subs@admin-magazine.com.

In This Issue

Welcome: A Thousand Words Paint a Picture

Every system administrator wants to automate the tedious and mundane tasks they must perform regularly. But there's a dark side to automation, too.

Photo by nick on Unsplash

FOSS
BACKSTAGE

Behind the scenes of Open Source projects

*10–11 MARCH 2025
IN BERLIN + ONLINE

Join our conference for two exciting days focused on governance, collaboration, InnerSource, OSPOs, project leadership, community management, and the legal and economics in Open Source.

Get your ticket
and save 10% with
the code LINUXMAG10

25.foss-backstage.de

HETZNER

HETZNER
OBJECT STORAGE

BUILT TO GROW WITH YOUR NEEDS

Manage data volumes simply and efficiently!

- ✓ S3 compatible storage solution
- ✓ High availability
- ✓ Durable data storage
- ✓ Scaleable
- ✓ Object Locking
- ✓ Data Protection
- ✓ No minimum contract
- ✓ Billing on an hourly basis
- ✓ NEW: Payment in US$

$ 5.99 | € 4.99

1 TB traffic & storage per month

All prices exclude VAT and are subject to the terms and conditions of Hetzner Online GmbH. Prices are subject to change. All rights reserved by the respective manufacturers.

Ideal solution for managing data-intensive workloads

Whether for backups, multimedia or big data

Start now:
htznr.li/linux/object-storage

SCAN THE CODE

www.hetzner.com